GUIDA AL SINCLAIR ZX81

ZX80 e Nuova ROM

GRUPPO
EDITORIALE
JACKSON

RITA BONELLI

GUIDA AL SINCLAIR

Z X 8 1

ZX80-NUOVA ROM

di
Rita Bonelli

GRUPPO
EDITORIALE
JACKSON
Via Rosellini, 12
20124 Milano

P R E F A Z I O N E

In seguito al successo di vendita del personal computer ultraeconomico ZX80 sono stati, successivamente, introdotti anche nel nostro paese prima una versione potenziata dello stesso tramite nuove memorie ROM e, adesso, il modello maggiore ZX81 sempre della casa inglese Sinclair.

La simpatia e la versatilita' del piccolo sistema sono certamente all'origine della sua fortuna, ma un buon contributo l'ha anche dato in Italia il manuale "Impariamo a programmare in BASIC con lo ZX80", ispirato com'era anche a principi di carattere educativo. Con una macchina cosi' "personale" oltre che alla portata di molte tasche e' infatti fondamentale - insieme ai dati costruttivi ed alle modalita' d'uso - un minimo d'insegnamento sulle regole del gioco programmatorio. Alla gente piace infatti che gli si dica quale puo' essere il modo migliore per utilizzare vantaggiosamente questi calcolatori in miniatura ma la cui potenza e' tutt'altro che indifferente sol che si sappia come sfruttarla al meglio. Tanto piu' che senza il software e, quindi, senza la capacita' di svilupparselo per lo piu' autonomamente (dato che acquistarlo, a questi bassi livelli di costo dell'hardware, e' cosa pressoche' priva di senso), tali oggetti non servono letteralmente a nulla.

Ma, come si e' detto in apertura, per venire incontro a maggiori necessita' la fisionomia - nel passare dal primitivo ZX80 a quello attrezzato con ROM da 8K anziche' 4K e, infine, all'odierno ZX81 - e' risultata modificata in taluni connotati: del sistema di gestione e del linguaggio Basic soprattutto, pur rimanendo praticamente immodificati la filosofia e l'impianto di fondo.

Che fare in queste condizioni? Anziche' scrivere un nuovo manuale in aggiunta al precedente si e' ritenuto di farne uno in sostituzione di quello. Una scelta che appare piu' che saggia, tenendo presente che la vecchia edizione non viene piu' stampata.

"Guida al Sinclair" risulta cosi' un testo completo che si rivolge ad utenti vecchi e nuovi. Il nucleo concettuale-formativo (frutto dell'esperienza didattica e professionale dell'Autrice, che ha gia' all'attivo diversi altri testi del genere) e' rimasto, anzi si offre ulteriormente arricchito dall'aver tenuto il piu' possibile conto di diverse osservazioni e richieste pervenute da parte dell'ormai abbastanza numerosa famiglia di utilizzatori Sinclair.

Dovendo poi parlare di tutte e tre le possibili configurazioni l'Autrice ne ha anche approfittato per operare tutti quei necessari raffronti relativi alle

differenze, mirando non solo ad indicare con la massima chiarezza a ciascuno i caratteri del suo modello e la relativa "lingua", ma anche spunti di riflessione in materia di possibilita' e limiti che ciascun contesto puo' presentare.

Anche da qui puo' cosi' derivare un piccolo ma significativo spunto a saper guardare un pochino al di la' del proprio "particolare".

Gianni Giaccaglini

S O M M A R I O

CAPITOLO 1

P R E M E S S E

1.1. INTRODUZIONE

Perche' questo manuale? Per soddisfare le richieste dei lettori. Sono stati venduti tanti SINCLAIR ZX80 e tanti relativi manuali. Poi e' arrivata la Nuova ROM per lo ZX80, ne sono state vendute tante insieme al relativo manualetto. Poi tanti lettori telefonano o scrivono per chiedere ulteriori delucidazioni; interessano le modalita' per trasformare i programmi da un calcolatore all'altro, si vuole sapere qualcosa sul Sistema Operativo, sul linguaggio macchina.

Ora arriva lo ZX81 e tante altre persone entreranno nel paese dell'informatica. L'Editore mi ha chiesto di fare un nuovo manuale ed io mi sono accinta all'impresa con piacere.

Perche' con piacere? Effettivamente puo' anche non essere considerato un divertimento scrivere tanti manuali sui piccoli calcolatori e sul Basic. Ma il piacere deriva dal fatto che io sono contenta che tanta gente impari ad usare un calcolatore. Ora che il costo di un personal, tipo SINCLAIR, e' diventato accessibile a molti, la cultura informatica si puo' diffondere; io mi rendo conto che si sta diffondendo. Non mi e' mai piaciuto essere considerata un po' speciale per il mestiere che faccio da molti anni. Ho sempre ritenuto che il mestiere dell'informatico non e' poi cosi' difficile! Basta cominciare ad occuparsene ed avere un calcolatore a disposizione. Il calcolatore e' infatti essenziale. Non si puo' imparare l'informatica solo sui libri, ci vuole anche una buona dose di pratica.

Inoltre i vantaggi dei personal sono molteplici. Il Sistema Operativo e' abbastanza semplice, l'approccio con il linguaggio Basic rende tutto abbastanza semplice, con un po' di pazienza e' possibile approfondire gli argomenti, arrivare a conoscere tutto del vostro calcolatore, arrivare al linguaggio macchina.

Il SINCLAIR vi da' molte possibilita' di apprendimento, sempre che la cosa vi interessi, vi appassioni e vi diverta.

Spero di aver contribuito con questa guida a mettervi

nelle condizioni di usare con piacere il vostro calcolatore.

Della guida fanno parte i due precedenti manuali ZX80 e Nuova ROM fusi e, spero, con eliminazione degli errori che erano inevitabilmente scappati. Sono inoltre presenti delle parti nuove che non esauriscono completamente gli argomenti piu' difficili, ma spero servano a risvegliare l'interesse dei lettori verso maggiori approfondimenti. Tramite le riviste specializzate della Jackson continuero' ad occuparmi della famiglia Sinclair cercando di completare argomenti non approfonditi del tutto e mettendo in luce altre possibilita' di questi piccoli ed interessanti calcolatori.

1.2. STRUTTURA DEL MANUALE

Il manuale descrive 3 calcolatori:

- ZX80;
- ZX80-Nuova ROM;
- ZX81.

A mio avviso e' molto interessante paragonare tra loro i diversi calcolatori e comprenderne le differenze.

Per coloro che desiderano cominciare a programmare in Basic e' sufficiente leggere ed operare in base ai primi 6 Capitoli del libro.

I Capitoli 7 e 8 sono per coloro che, dopo aver appreso a programmare bene in Basic desiderano proseguire verso mete piu' lontane, anche se raccomando il Capitolo 7 anche a coloro che vogliono solo imparare a programmare in Basic.

Nel Capitolo 9 sono contenuti parecchi programmi utili per tutti e adatti ai diversi calcolatori. In questo stesso capitolo si parla dei file e si toccano argomenti molto interessanti come l'animazione delle figure sui tre calcolatori.

Le Appendici da A ad E interessano tutti a seconda delle diverse esigenze. Le Appendici F, G e H riguardano il linguaggio macchina e i due Sistemi Operativi e quindi sono interessanti per coloro che vogliono approfondire le loro conoscenze informatiche.

In qualche punto potrete trovare delle ripetizioni, esse sono volute e penso che facilitino il lettore in particolari momenti del suo lavoro.

CAPITOLO 2

I L   C A L C O L A T O R E

2.1. STRUTTURA DEL CALCOLATORE

Le parti componenti un calcolatore elettronico, vedi Fig. 2.1., sono in generale le seguenti:

- . unita' centrale (CPU);
- . unita' di ingresso (INPUT);
- . unita' di uscita (OUTPUT);
- . memoria secondaria.

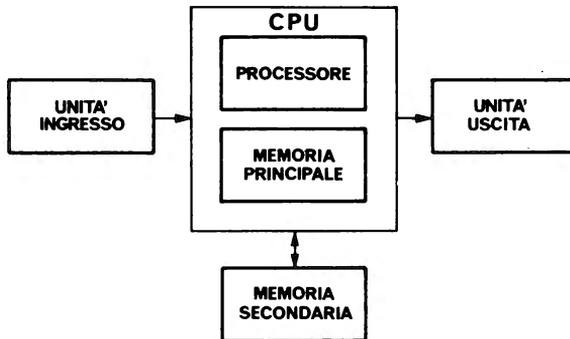



Fig. 2.1. Struttura del calcolatore

Una elaborazione con il calcolatore consiste sempre in una trasformazione di dati. I dati di ingresso vengono elaborati dal calcolatore e trasformati nei dati di uscita.

Le parti componenti il SINCLAIR sono:

- . unita' centrale CPU;
- . unita' di ingresso, che e' una tastiera sensibile al tocco.

L'unita' di uscita e' un qualunque schermo TV (la televisione di casa) e la memoria secondaria e' una cassetta magnetica su un registratore (quello di casa).

L'unita' centrale del calcolatore e' formata da:

. microprocessore Z80A con clock a 3.25 MHz;
. memoria a sola lettura, ROM (Read Only Memory);
. memoria per lettura e scrittura, RAM (Random Access Memory).

Le dimensioni della memoria vengono date usando la costante K che e' uguale a 1024 ed il nome BYTE che significa UNITA' DI MEMORIA. Spesso il nome BYTE viene omesso. Per distinguere i diversi byte costituenti la memoria si usa un indirizzo numerico che parte da zero. Il BYTE e' la piu' piccola parte di memoria che puo' essere indirizzata.

Lo ZX80 ha una memoria ROM di 4K ed una memoria RAM standard di 1K, estendibile fino a 16K. La nuova ROM, montabile sullo ZX80, e' di 8K. Lo ZX81 ha una memoria ROM di 8K ed una memoria RAM standard di 1K estendibile fino a 16K.
La memoria ROM non puo' essere scritta dall'utente; essa contiene in forma stabile il corredo di programmi necessari per il funzionamento del calcolatore.
La memoria RAM serve per memorizzare i programmi scritti dall'utente, i dati ed i risultati, ma essa e' labile, cioe' si cancella quando l'utente lo desidera e comunque quando si spegne il calcolatore.
Per questa ragione si usa la memoria secondaria, costituita dalla cassetta magnetica, per registrare programmi e dati in modo permanente.

Il microprocessore comprende:

. unita' di governo, che controlla lo svolgimento delle istruzioni del programma;
. unita' aritmetico/logica, che esegue le operazioni aritmetiche e i controlli logici;
. alcuni registri speciali, usati come memoria di lavoro dal microprocessore.

Ogni calcolatore nasce con la capacita' di svolgere un gruppo finito di istruzioni, tale gruppo di istruzioni costituisce il LINGUAGGIO MACCHINA DEL CALCOLATORE. Una opportuna sequenza di istruzioni in linguaggio macchina costituisce un PROGRAMMA per il calcolatore.
Il programma si memorizza nella memoria del calcolatore e l'unita' centrale, opportunamente avviata, e' capace di prelevare automaticamente le istruzioni del programma dalla memoria e di eseguirle una dopo l'altra.
Nella Fig. 2.2. e' riportato il calcolatore ZX81 aperto; se si confronta con il capostipite ZX80 si vede che il numero dei componenti e' diminuito.

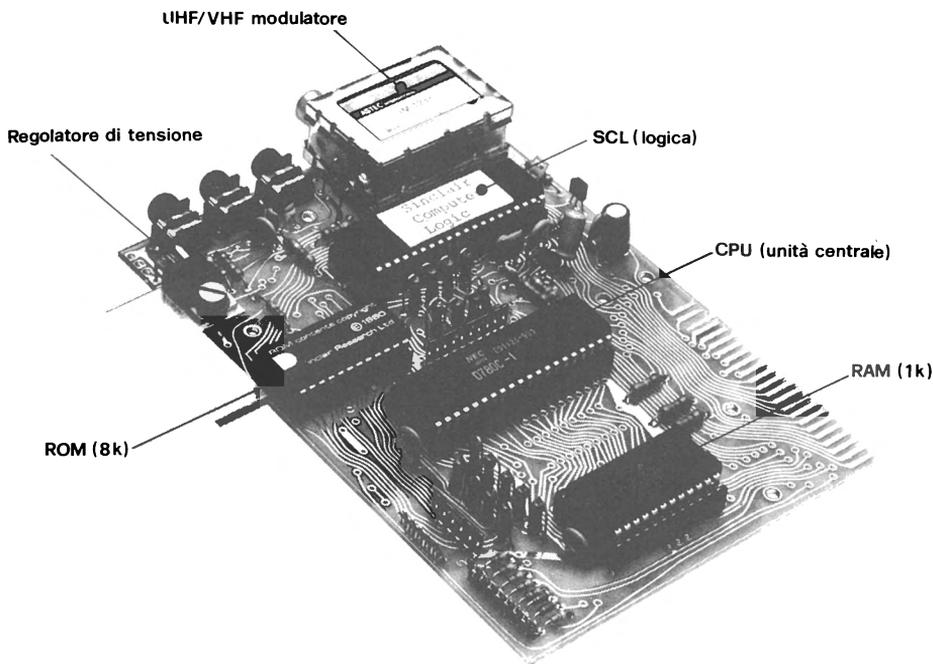



Fig. 2.2. Il calcolatore ZX81 aperto

2.2. LA MEMORIA PRINCIPALE

La memoria principale, sia ROM che RAM, e' formata da un certo numero di byte contraddistinti da un numero che costituisce il loro indirizzo. Gli indirizzi partono da 0. Ogni microprocessore ha la possibilita' di indirizzare byte fino ad un valore massimo; il SINCLAIR nella configurazione attuale puo' indirizzare fino a 32767. A seconda della RAM utilizzata, 1K o piu', sono accessibili piu' o meno indirizzi.

La memoria puo' essere immaginata come costituita da una serie di cellette contigue; esse sono i byte.

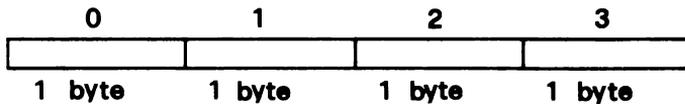



Fig. 2.3. Schema della memoria

Nella memoria le informazioni sono registrate in forma binaria, cioe' di numeri le cui cifre possono essere solo 0 e 1. Nei numeri binari il valore posizionale delle cifre si calcola in base alle potenze di 2. Un byte puo' contenere 8 cifre binarie; ogni cifra binaria viene chiamata BIT. I singoli bit non sono indirizzabili; essi sono indirizzabili solo a gruppi di 8, infatti 8 bit costituiscono 1 byte.

Da quanto detto sopra si deduce che qualunque informazione entra nel calcolatore in codice numerico binario; fortunatamente l'utente puo' usare i caratteri a lui gia' noti, numeri decimali, lettere e caratteri speciali e pensano alcuni programmi della ROM a operare la trasformazione.

Nel Capitolo 7 viene descritto l'uso della memoria da parte del sistema.

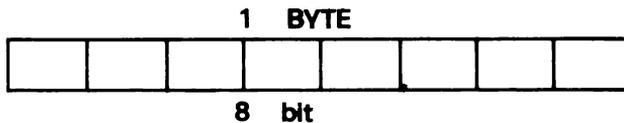



Fig. 2.4. Byte e Bit

2.3. L'AUTOMATISMO DEL CALCOLATORE

Il calcolatore e' una macchina automatica, cioe' una macchina che, dopo essere stata avviata funziona da sola. L'automatismo del calcolatore consiste in questo:

. le istruzioni per il calcolatore devono essere memorizzate in un gruppo di byte consecutivi della memoria partendo da un certo indirizzo;

. l'indirizzo della prima istruzione da eseguire deve essere posto in un registro speciale che prende di solito il nome di Contatore del Programma;

. si deve dare al calcolatore il comando di avvio, che di solito consiste nella pressione di un particolare tasto;

. il calcolatore preleva dall'indirizzo di memoria contenuto nel Contatore l'istruzione da eseguire e la

trasferisce in un registro speciale dedicato alla esecuzione delle istruzioni e contemporaneamente incrementa il contenuto del Contatore (in tale modo il contatore viene a contenere l'indirizzo della prossima istruzione da eseguire);

. il calcolatore esegue l'istruzione ed al termine di questa ritorna al punto precedente.

E' evidente che il calcolatore porta avanti questo automatismo fino a quando interviene qualcosa a fermarlo. Questo qualcosa puo', per esempio, essere l'esecuzione della istruzione STOP.

Esistono tante altre cause che possono fermare il lavoro del calcolatore, alcune sono anche un po' complicate da comprendere e quindi non e' il caso di parlarne ora.

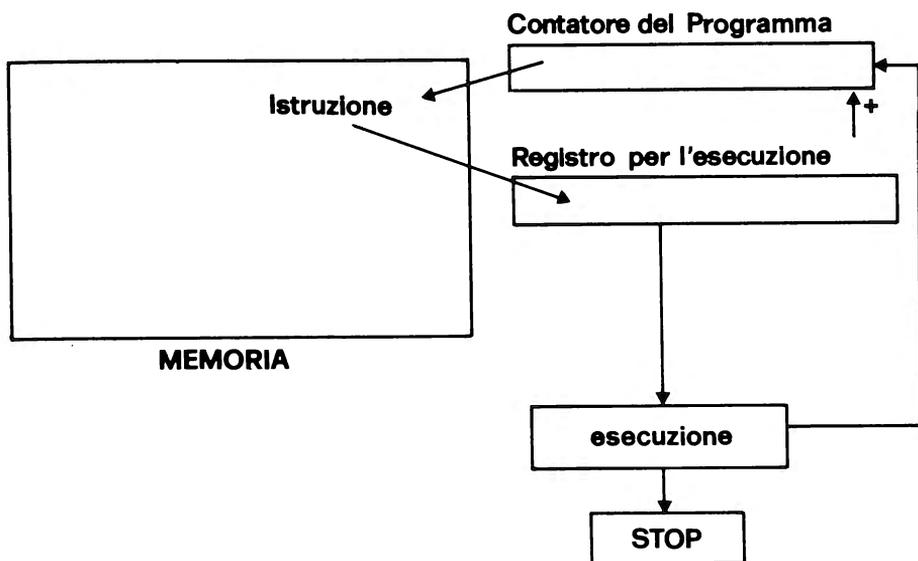



Fig. 2.5. Schema dell'automatismo

Nel Sinclair non esiste un tasto per l'avvio del calcolatore in linguaggio macchina.

2.4. IL SISTEMA OPERATIVO

Ogni calcolatore e' in generale dotato di un corredo di programmi che vengono forniti insieme al calcolatore e che

ne facilitano l'uso. Questo non e' indispensabile, nel senso che si potrebbe usare felicemente anche un calcolatore privo di programmi base, ma sarebbe piu' lungo e difficile pervenire a dei risultati. Inoltre il singolo utente dovrebbe rifare un grosso lavoro, che tutto sommato e' standardizzabile e quindi puo' essere fatto a priori dalla casa costruttrice.

Ricordando l'automatismo di funzionamento del calcolatore si comprende che per far funzionare il calcolatore basta saper mettere insieme una serie di istruzioni in linguaggio macchina, scriverle in memoria ed avviare il processo automatico.

La stesura di programmi in linguaggio macchina risulta abbastanza difficile; per questa ragione sono stati messi a punto dei linguaggi di programmazione di facile apprendimento per l'uomo, e si e' pensato di fare svolgere al calcolatore il lavoro di traduzione da tali linguaggi in linguaggio macchina.

Questo lavoro di traduzione e' necessario dato che il calcolatore capisce solo il suo linguaggio macchina.

Inoltre si e' cercato di corredare il calcolatore di tutti quei programmi che ne facilitano l'uso, cioe' che rendono piu' semplice scrivere nella memoria del calcolatore, leggere dalla memoria, scrivere sul nastro magnetico, ecc.

L'insieme di questi programmi costituisce il SISTEMA OPERATIVO del calcolatore. Per il Sinclair il Sistema Operativo e' gia' registrato nella memoria ROM e quindi sta perennemente dentro il calcolatore. Se si apre il calcolatore e si sostituisce la ROM si puo' disporre di un nuovo Sistema Operativo.

Fortunatamente per l'utente, dato che risiede in ROM, il Sistema Operativo non puo' essere distrutto commettendo errori nell'uso del calcolatore.

Nelle Appendici G e H si trovano utili informazioni sulle 2 versioni del Sistema Operativo.

2.5. IL VIDEO

Quando il vostro sistema e' acceso sul video compare su sfondo chiaro nell'angolo in basso a sinistra un quadratino piu' scuro lampeggiante con al centro una lettera K piu' chiara. Questo quadratino si chiama CURSORE dello schermo. La lettera che compare al centro del cursore indica lo stato nel quale si trova il calcolatore. I caratteri possono apparire sul video scuri su fondo chiaro e questo avviene di norma, oppure chiari su campo scuro, cioe' in campo inverso.

Il video puo' contenere 24 linee di 32 caratteri ciascuna.

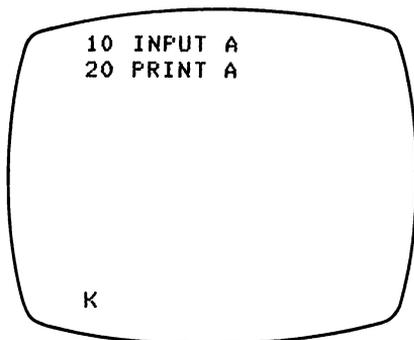



Fig. 2.6. Il video

Quando il cursore e' nello stato K il calcolatore e' in attesa di comandi.
Oltre al cursore, sullo schermo si ha un altro indicatore il PUNTATORE DI LINEA; esso e' rappresentato da un quadratino scuro con in chiaro al centro il simbolo di maggiore (>). Normalmente questo puntatore segna l'ultima linea di programma scritta durante il caricamento di un programma.

Durante l'introduzione di un programma il cursore lavora nella parte bassa dello schermo e segue quello che voi scrivete. Esso puo' essere spostato usando i tasti freccia-a-sinistra (SHIFT e 5) e freccia-a-destra (SHIFT e 8). Quando la linea di programma viene accettata essa si sposta nella parte alta dello schermo e viene puntata dal puntatore di linea. Il cursore dello schermo resta in basso.

Il puntatore di linea puo' analogamemnte essere spostato usando i tasti freccia-in-su (SHIFT e 7) e freccia-in-giu' (SHIFT e 6).

Vediamo ora i possibili stati del calcolatore separatamente per lo ZX80, lo ZX80-Nuova ROM e lo ZX81.

Per lo ZX80 gli stati possibili sono due; il calcolatore puo' essere nello stato K di attesa comandi oppure nello stato L. Se sul cursore compare L questo significa che il calcolatore e' in attesa di caratteri.
Inoltre il cursore si sdoppia, cioe' compaiono due cursori, in caso di errore o di attesa di dati numerici. In questo caso il nuovo cursore contiene la lettera S (errore Sintassi). Il cursore dello schermo, sdoppiato in caso di errore, si pone con la parte S prima dell'errore e la parte

L dopo l'errore. Nel caso di attesa di dato numerico le due parti stanno vicine con L prima di S.

Quando un programma lavora e vengono incontrate operazioni di INPUT (ingresso dati) il cursore si pone nella parte alta dello schermo alla prima linea libera e segnala l'attesa di un numero con LS, come detto prima, e l'attesa di una stringa con "L".

Anche il tasto HOME (SHIFT e 9) agisce sul puntatore di linea facendolo salire alla linea zero. Dal momento che la linea zero non esiste sullo schermo, usando HOME, il puntatore di linea svanisce. Se si vuole far apparire di nuovo il puntatore di linea basta usare il tasto freccia-in-giu' (SHIFT e 6).

Per lo ZX81 e lo ZX80-Nuova ROM gli stati possibili sono quattro:

. stato K di attesa comandi;
. stato L di attesa carattere;
. stato G di attesa carattere grafico;
. stato F di attesa funzione.

Lo stato F resta attivo solo per l'introduzione di una singola funzione. Lo stato G resta attivo fino a quando non lo si elimina premendo di nuovo SHIFT e GRAPHICS. Quando il cursore segna lo stato K il calcolatore e' in attesa di comandi. Lo stato L significa attesa di dati. Lo stato G significa attesa di caratteri grafici e lo stato F attesa di un comando funzione. Gli stati K ed L sono prodotti automaticamente dal Sistema Operativo, mentre gli stati G ed F sono comandati dall'utente.

Qui non si ha lo sdoppiamento del cursore quando si e' in attesa di INPUT, ed inoltre il cursore resta nella parte bassa dello schermo quando e' in attesa di dati. Lo stato L significa attesa di dati numerici; se il cursore appare con L tra apici ("L") significa che attende una stringa. Se si risponde con una stringa all'attesa di dati numerici si ha segnalazione di errore 2, il calcolatore non accetta il dato, ma scrivendo CONT si puo' continuare introducendo di nuovo il dato corretto.

In fase scrittura programma la segnalazione dell'errore compare quando si cerca di fare accettare la linea con NEW LINE; in questo caso il cursore si sdoppia e la parte con S (errore Sintassi) sta sinistra dell'errore quella con L a destra.

In questa versione non esiste il tasto HOME. Il puntatore di linea puo' salire al massimo fino alla prima linea di programma usando il tasto freccia-in-su o dando il comando LIST senza il numero di linea.

2.6. LA TASTIERA ZX80

Osservando la tastiera (Fig. 2.7.) si vede che alcuni tasti hanno una sola funzione, scritta in bianco all'interno, mentre sopra il tasto e' riportata una parola o un simbolo grafico. A questo gruppo appartengono i tasti: 1,2,3,4,5,6,7,8,9,0 e NEW LINE.

Per attivare la funzione scritta sopra il tasto, in questo caso si deve tenere premuto il tasto SHIFT. Il tasto SHIFT ha una sola funzione: attivare lo SHIFT.

Quasi tutti gli altri tasti hanno due funzioni scritte all'interno, una in bianco e una in giallo, ed inoltre una funzione scritta sopra il tasto.

Se il calcolatore e' nello stato K, rilevabile dal cursore in campo inverso dello schermo, premendo un tasto senza SHIFT si attiva la funzione scritta sopra; mentre se il calcolatore e' nello stato L, rilevabile sempre dal cursore dello schermo, premendo un tasto senza SHIFT si attiva la funzione scritta in bianco all'interno del tasto. La funzione scritta in giallo all'interno del tasto, si attiva, per questo gruppo, premendo lo SHIFT contemporaneamente al tasto.

Per usare la tastiera il movimento delle dita deve essere delicato ed i tasti non devono essere battuti come sulle macchine da scrivere. E' importante imparare a distinguere la lettera O dallo zero. Sulla tastiera lo zero si trova in alto a destra dopo il 9 ed e' meno rotondo della lettera O che si trova nella fila sotto.

Per ottenere i caratteri in campo inverso si deve usare la funzione CHR$ con il codice ASCII del carattere voluto; questi caratteri non sono ottenibili da tastiera.

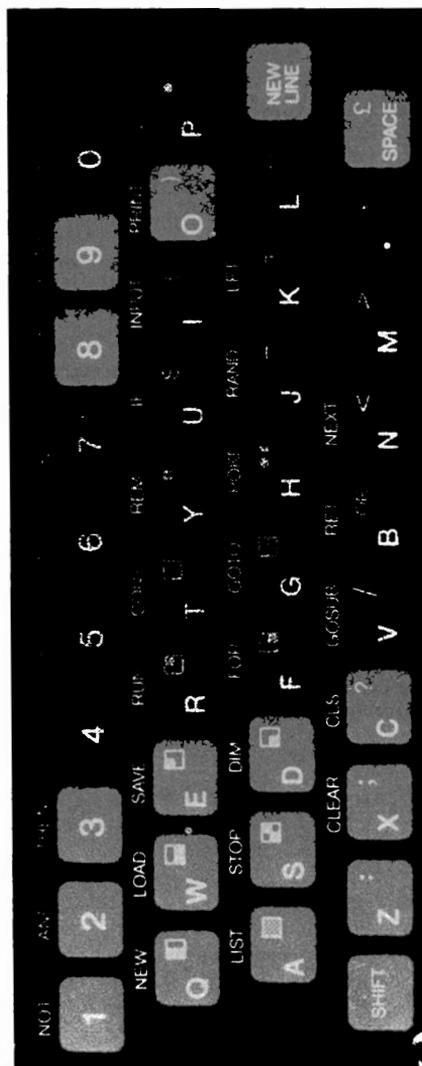



Fig. 2.7. La Tastiera dello ZX80

2.7. LA TASTIERA ZX81

Questa tastiera e' quella fornita insieme alla nuova ROM per sostituirla nello ZX80, ed e' anche quella dello ZX81.
Nella nuova tastiera (Fig. 2.8.) solo il tasto SHIFT reca una sola dicitura; esso serve:

. per attivare le funzioni scritte in rosso sugli altri tasti;
. per cambiare lo stato del calcolatore (G e F);
. per ottenere i caratteri grafici;

e l'effetto prodotto dipende dallo stato nel quale si trova il calcolatore. Tale stato e' sempre rilevabile dal carattere evidenziato in campo inverso sul cursore dello schermo.
Gli altri tasti hanno tutti piu' funzioni e queste vengono rese attive, sempre in dipendenza dalla stato del calcolatore, senza o con l'uso contemporaneo del tasto SHIFT. Esaminiamo cio' che e' scritto sui tasti. Abbiamo:

. cifre, lettere, simboli o caratteri grafici in nero nella parte bassa;
. simboli o parole in rosso nella parte alta.

Le cifre, le lettere e i simboli vengono accettati quando il cursore dello schermo si trova nello stato L.
I caratteri grafici sono accettati quando il cursore si trova nello stato G (si passa a questo stato premendo contemporaneamente SHIFT e GRAPHICS) e si premono contemporaneamente il tasto SHIFT e il tasto del carattere grafico che interessa.
Se il cursore si trova nello stato G e si preme un qualunque tasto, senza usare lo SHIFT, si ottiene il carattere (non grafico) in campo inverso.
Per uscire dallo stato G e tornare allo stato L si devono ancora premere contemporaneamente SHIFT e GRAPHICS.
Se il calcolatore e' in attesa di stringa e si passa allo stato G per fare accettare la stringa si deve ritornare allo stato L e premere NEW LINE o premere 2 volte NEW LINE.
I simboli e le parole in rosso vengono accettati se si preme il tasto contemporaneamente allo SHIFT, qualora il cursore indichi lo stato L.
Le parole scritte sotto i tasti sono considerate funzioni e sono attive quando il cursore indica lo stato F. Lo stato F si ottiene premendo contemporaneamente i tasti SHIFT e FUNCTION.
Le parole scritte sopra i tasti sono parole chiave del linguaggio BASIC e sono attive quando il cursore indica lo stato K.
I comandi FAST e SLOW sono validi solo per lo ZX81.

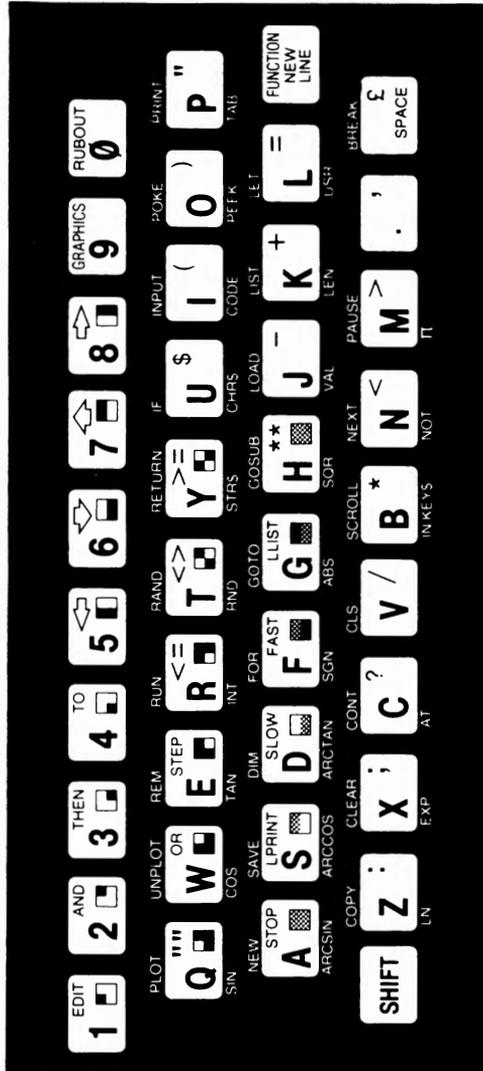



Fig. 2.8. La tastiera dello ZX81

2.8. LE PERIFERICHE

Si possono aggiungere delle espansioni di memoria RAM, inserendole nella fessura larga che si trova dietro il calcolatore (a sinistra nello ZX80 e a destra nello ZX81). Sono disponibili espansioni da 3K e da 16K. Inserendo l'espansione da 3K il calcolatore viene ad avere disponibile una memoria RAM da 4K. Inserendo invece l'espansione da 16K si annulla, per cosi' dire, 1K di memoria standard presente nel calcolatore e restano attivi i 16K aggiunti.

Oltre al video, che e' indispensabile per poter usare il calcolatore, e' quasi altrettanto indispensabile collegare un registratore al SINCLAIR. Infatti senza registratore non si possono conservare i programmi che si scrivono e i dati che si elaborano.

Il registratore puo' essere di qualunque tipo, sia a bobina che a cassette, sia stereofonico che monosonico. L'unica condizione necessaria e' che il registratore sia dotato di un ingresso per microfono separato e di una uscita per auricolare o cuffia.

Con il nuovo BASIC, quello disponibile sullo ZX80-Nuova ROM e sullo ZX81, si puo' collegare una stampante al calcolatore. Essa e' stata progettata apposta per il SINCLAIR, consente di stampare su 32 colonne e consente di fare della grafica molto sofisticata. Inoltre e' possibile trasferire sulla stampante il contenuto del video in qualunque momento. Nella Fig. 2.9. si riporta la stampante e nella Fig. 2.10. un esempio di listato di programma. La stampante si collega tramite la larga fessura posta sul retro ed al connettore della stampante si collega l'espansione della memoria.

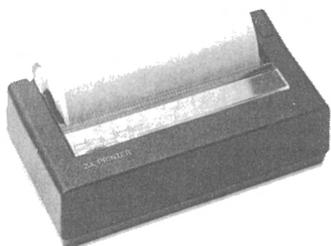



Fig. 2.9. La stampante

```
10 INPUT A$
20 PRINT A$
30 LPRINT A$
40 GOTO 10

PRIMA RIGA
SECONDA RIGA
TERZA RIGA
```

Fig. 2.10. Listato programma

2.9. IL LINGUAGGIO MACCHINA

Il SINCLAIR puo' anche essere programmato in linguaggio macchina, e questo e' consigliabile per approfondire la conoscenza del calcolatore. Nel Capitolo 8 si descrivono le procedure per poter passare dal BASIC al linguaggio macchina e nella Appendice F sono riportate le istruzioni del linguaggio macchina.
Il Sinclair e' stato costruito per colloquiare in Basic; per questa ragione anche se si programma in linguaggio macchina, i programmi vanno introdotti usando il Basic. Inoltre anche per mandare in esecuzione un programma in linguaggio macchina e' necessario servirsi del Basic.

2.10. IL LINGUAGGIO BASIC

Il linguaggio BASIC e' un linguaggio simbolico ad alto livello di tipo interpretativo. Questo significa che quando si usa, in gergo "si fa girare", un programma scritto in BASIC, nella memoria del calcolatore deve anche essere presente un programma (ovviamente scritto in linguaggio macchina), chiamato INTERPRETE BASIC, che ha il compito di tradurre le frasi del linguaggio BASIC in istruzioni in linguaggio macchina eseguibili dal calcolatore. L'utente non si accorge di questo grosso lavoro che compie il sistema, ma questo lavoro viene svolto. Il programma interpretatore risiede nel SINCLAIR nella memoria ROM, insieme al Sistema Operativo. Nei due calcolatori, ZX80 e ZX81 (oppure ZX80-Nuova ROM) si hanno due ROM diverse e quindi si hanno differenze sia a livello di Sistema Operativo che di BASIC.

Si e' definito il BASIC come "linguaggio simbolico ad alto livello"; questo significa che il programmatore lavora con dei nomi simbolici, per lui di piu' facile comprensione, e che ogni istruzione o, come si suole anche dire, frase del linguaggio, corrisponde ad un bel gruppo di istruzioni in linguaggio macchina. In tale modo viene implicitamente definito a "basso livello" il linguaggio macchina. Con questo non si vuole assolutamente declassare il linguaggio macchina, che resta, per eccellenza, il linguaggio degli specialisti e degli appassionati dei calcolatori. Solo che i calcolatori sono degli strumenti di lavoro che sempre di piu' si diffondono nella societa' moderna, ed e' quindi necessario che possano essere usati da tutti e non solo dagli specialisti. Un linguaggio come il BASIC, estremamente facile e comprensibile, ha molto favorito la diffusione dei calcolatori fra la gente.

L'interpretazione giusta da dare alle parole "basso" ed

"alto" livello e' la seguente:

. nei linguaggi a basso livello una istruzione scritta nel codice proprio del linguaggio corrisponde ad una sola istruzione in linguaggio macchina;

. nei linguaggi ad alto livello ad una istruzione scritta corrispondono piu' istruzioni in linguaggio macchina.

E' molto importante per l'utente fare la doppia esperienza del vecchio e nuovo Basic del SINCLAIR, potra' in tale modo vedere che la filosofia del linguaggio e' sempre la stessa anche se nelle diverse versioni (che in gergo si dicono "implementazioni") si riscontrano alcune differenze.

2.11. LE DIFFERENZE TRA I CALCOLATORI SINCLAIR E IL BASIC STANDARD

Le differenze tra lo ZX80 e, lo ZX80-Nuova ROM e ZX81, dipendono dal fatto che nel primo calcolatore si ha una ROM di solo 4K con una versione ridotta del Basic ed un Sistema Operativo adeguato. La ROM degli altri due calcolatori e' di 8K ed e' disponibile una nuova versione di Basic con un nuovo Sistema Operativo. L'unica differenza che si ha tra lo ZX80-Nuova ROM e lo ZX81 consiste nel fatto che in quest'ultimo e' attiva da tastiera la funzione FAST/SLOW. Questa funzione, se rende attivo il modo SLOW, fa si che non scompaiano le scritte dallo schermo mentre il calcolatore lavora. Questo naturalmente rende meno veloce il calcolatore (di circa 1/4), ma consente di ottenere una grafica migliore e l'animazione delle figure sul video.
Lo ZX80 e lo ZX80-Nuova ROM lavorano sempre in modo FAST; in tale modo puo' naturalmente lavorare anche lo ZX81.

Le piu' vistose differenze tra le due implementazioni del Basic, viste dalla parte della ROM da 8K, sono le seguenti:

- sono disponibili i numeri decimali;
- sono disponibili molte funzioni in piu';
- sono disponibili le variabili stringa con indice;
- e' possibile gestire indici multipli;
- sono disponibili nuove istruzioni per la grafica;
- cambia il significato degli operatori logici;
- si possono trattare parti di stringa;
- si puo' collegare una stampante;
- si possono memorizzare i programmi con un nome.

Nel corso del manuale verranno messe in evidenza tutte le caratteristiche dei due linguaggi e si faranno continuamente degli interessanti confronti.

Le differenze rispetto ai Basic piu' comuni (detti standard) possono essere sintetizzate da quanto segue.

Nello ZX80 sono disponibili solo i numeri interi, con il nuovo Basic sono disponibili solo i numeri decimali. Si hanno delle differenze nella definizione delle variabili con indice.

Non e' disponibile il comando:

```
ON X GOTO N1,N2,N3,...NK
```

si puo' ottenere lo stesso risultato usando alcuni accorgimenti. Invece di scrivere:

```
ON X GOTO 100,200,300,400
```

che ha il significato di mandare: alla linea 100 se X=1,
" " 200 " X=2,
" " 300 " X=3,
" " 400 " X=4;

si puo' scrivere:

```
GOTO 100*X
```

e si ottiene lo stesso risultato.

Non sono disponibili i comandi: READ, DATA e RESTORE per gestire blocchi di dati all'interno di un programma. Ricordiamo che la DATA serve per memorizzare blocchi di dati all'interno di un programma, la READ serve per associare questi dati alle variabili in sequenza e la RESTORE serve per poter ricominciare ad usare i dati dall'inizio del blocco.

Si puo' ottenere il risultato di avere un gruppo di variabili con determinati contenuti operando in diversi modi:

.1) Scrivere una serie di LET variabile = dato.
.2) Scrivere una serie di istruzioni di lettura dati dal= l'esterno all'inizio del programma, eventualmente con un ciclo FOR se i nomi delle variabili lo consentono, e poi memorizzare il programma su nastro insieme alle variabili (si vedano i paragrafi 4.8. e 9.14.).
.3) Incorporare i dati in delle REM o in delle stringhe lunghe e poi usare delle routine di smistamento dei dati.

Resta sempre la limitazione sull'uso dei file di dati, non

gestibili direttamente.

Per poter gestire direttamente file di dati su nastro deve essere possibile avviare e fermare da programma il registratore. Questo ora non e' possibile sui calcolatori Sinclair. Attualmente il registratore deve essere avviato manualmente e puo' solo scrivere o leggere un intero programma, comprese le sue variabili.

Nel Capitolo 9 si riportano alcuni programmi esempio che mostrano come si possa superare questa difficolta'.

CAPITOLO 3

I N S T A L L A Z I O N E D E L C A L C O L A T O R E

3.1. INSTALLAZIONE DELLO ZX80

Lo ZX80 e' composto da due unita':

. 1) il calcolatore;
. 2) l'alimentatore.

L'alimentatore deve fornire 9 Volts in corrente continua a 600 mA non stabilizzati. Il cavo di collegamento termina con uno spinotto Jack del diametro di 3,5 mm, col positivo collegato alla punta. Si osservi il diagramma della Fig. 3.1. che riporta i collegamenti.

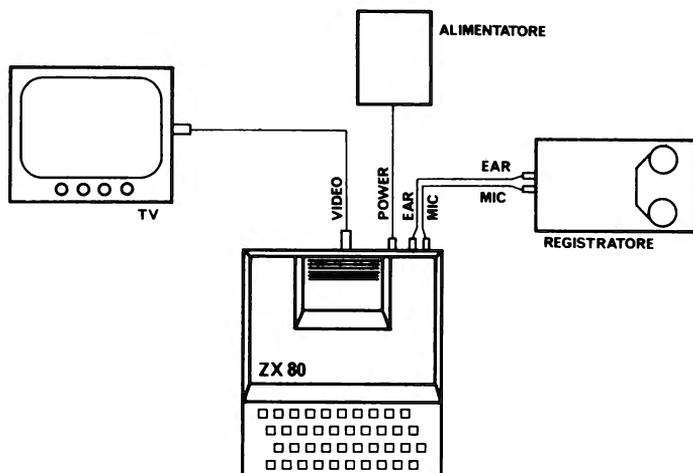



Fig. 3.1. Schema di collegamento per lo ZX80

Guardando il retro del calcolatore , Fig. 3.1., si vedono da sinistra a destra 3 prese nere per spinotti Jack, la cui nomenclatura e' riportata al di sotto del calcolatore. Il loro utilizzo e':

. TO RECORDER MIC, ingresso microfono del registratore;
. TO RECORDER EAR, uscita cuffia del registratore;
. 9 V DC IN, spinotto Jack dell'alimentatore.

Proseguendo verso destra, si vede in centro una presa per spinotto Plug americano, destinato al collegamento del Video.

Ancora piu' a destra si vede una larga fessura destinata all'inserimento della memoria aggiuntiva.

Come video puo' essere usato un qualunque apparecchio televisivo, sia in bianco e nero che a colori. Si selezioni la banda UHF (quella del secondo canale) e si sintonizzi il canale 36. Si abbassi il volume al minimo, dato che non esistono uscite sonore. L'uscita sul video e' predisposta per dare un quadro di 24 linee di 32 caratteri ciascuna. Si colleghi, utilizzando il cavo in dotazione, l'uscita video dello ZX80 con l'ingresso dell'antenna del televisore. Nel caso il televisore abbia due ingressi a doppio spinotto per l'antenna, sara' necessario munirsi di un adattatore di impedenza 75/300 Ohm e di cavo adeguato, con relativi spinotti, e collegarlo all'ingresso UHF.

A questo punto si accenda il televisore e, quando questo si e' scaldato, dopo aver inserito lo spinotto dell'alimentatore (con attenzione!) nella presa giusta (9 V DC IN), si accenda lo ZX80 collegando l'alimentatore alla rete. Quindi si aggiusti la sintonia fino a vedere lo schermo tutto bianco (o grigio chiaro) con nell'angolo a sinistra in basso un quadratino nero (CURSORE) contenente la lettera K in bianco. L'immagine deve essere assolutamente stabile. In caso l'immagine non sia buona, si provi a regolare la luminosita' ed il contrasto del televisore, ed a sintonizzare il quadro. La lettera K all'interno del quadratino nero deve essere chiaramente visibile.

Ora il calcolatore ZX80 e' in grado di funzionare. Si puo' eseguire il TEST che segue per controllare il corretto funzionamento del calcolatore. Si premano i tasti nella sequenza indicata e si controllino i risultati sullo schermo. I 5 richiami (* n) riguardano la prova dello ZX81 e dello ZX80-Nuova ROM e quindi il prossimo paragrafo.

PROGRAMMA PER IL CONTROLLO DEL CALCOLATORE

TASTO	SIGNIFICATO
1	Il cursore rimane K in campo inverso ed entra il numero 1
F	Dato che il cursore era in stato K, entra la

	parola FOR (quella scritta sopra il tasto) seguita da uno spazio ed il cursore passa allo stato L.
I	Dato che il cursore e' nello stato L entra la lettera I.
SHIFT + L	Tenendo premuto il tasto SHIFT, mentre si preme il tasto L, entra il carattere =.
1	Entra il numero 1.
SHIFT + 4	Tenendo premuto il tasto SHIFT, mentre si preme il tasto 4, entra TO (parola scritta sopra il tasto) seguito da uno spazio.
9	Entra il numero 9.
NEW LINE	Quanto scritto nella parte bassa dello schermo viene accettato come linea 1 di programma e va nella parte alta del video. Il cursore torna a evidenziare K.
2	Entra il numero 2.
O	Entra PRINT (parola scritta sopra il tasto della lettera O) seguito da uno spazio ed il cursore passa allo stato L. (* 1)
I	Entra il carattere I.
SHIFT + .	Tenendo premuto SHIFT entra il carattere virgola (quello situato in alto a destra sul tasto).
NEW LINE	La linea 2 viene accettata e sale in alto.
3	Il cursore era ritornato nello stato K, entra 3.
N	Entra NEXT seguito da uno spazio.
I	Il cursore era a L, entra il carattere I.
NEW LINE	La linea 3 viene accettata e sale in alto. Ora sullo schermo vedete: 1 FOR I = 1 TO 9 2 PRINT I, 3 NEXT I
R	Il cursore era tornato a K ed entra RUN.
NEW LINE	Per effetto di questo tasto viene accettato il comando RUN e viene eseguito il programma che e' stato appena scritto. Sullo schermo appaiono i

numeri da 1 a 9 in quattro colonne. In basso a destra compare 0/3 ad indicare che il programma ha terminato la sua esecuzione alla linea 3 con codice di errore 0, cioe' senza errori.
Premendo un tasto qualunque, appare la lista del programma ed il puntatore di linea alla linea 3. (* 2)

SHIFT + 7 — Muove il puntatore di linea in su.

SHIFT + 6 — Muove il puntatore di linea giu'.

SHIFT + 7 — Fa ritornare il puntatore di linea alla linea 2.

SHIFT + NEW LINE — Appare una copia della linea 2 in basso sullo schermo, con il cursore dello schermo situato dopo il numero di linea e la linea puo' essere modificata. (* 3)

SHIFT + 8 — Sposta il cursore verso destra di un carattere o di una parola chiave.

SHIFT + 8 — Sposta il cursore dopo la virgola.

SHIFT + 0 — Cancella il carattere a sinistra del cursore.

SHIFT + 5 — Sposta il cursore a sinistra di I.

2 — Inserisce il numero 2

SHIFT + P — Inserisce l'asterisco tra 2 e I. (* 4)

NEW LINE — Fa accettare la nuova versione della linea 2 al posto della vecchia. Ora sullo schermo appare:

```
1 FOR I = 1 TO 9
2 PRINT 2 * I
3 NEXT I
```

R — Entra la parola chiave RUN.

NEW LINE — Fa eseguire la nuova versione del programma e sullo schermo appaiono in colonna i numeri pari da 2 a 18 con ancora 0/3 in basso a sinistra.

Q — Fa entrare il comando NEW. (* 5)

NEW LINE — Fa eseguire il comando NEW, lo schermo viene ripulito, viene ripulita anche la memoria e il vostro programma non esiste piu'.

Facendo la prova precedente avete scritto il primo programma Basic per il vostro ZX80, l'avete eseguito, l'avete modificato ed avete eseguito il nuovo programma.

Si puo' procedere ora al collegamento del registratore, per completare il sistema. Puo' essere impiegato un qualunque tipo di registratore purche' sia presente un ingresso apposito per microfono ed una uscita per auricolare o cuffia. In dotazione si ha un doppio cavetto con 4 spinotti Jack di diametro 3,5 mm. Questo cavetto puo' essere usato per collegare lo ZX80 al registratore. Si colleghi l'uscita MIC dello ZX80 con l'ingresso per microfono (marcato MIC o REC) sul registratore e l'entrata EAR dello ZX80 con l'uscita per auricolare (marcata EAR o MONITOR) del registratore. E' importante familiarizzarsi con questi collegamenti perche' durante l'uso del registratore andranno fatti e disfatti piu' volte con sicurezza.

Se il registratore non ha l'ingresso per il microfono e l'uscita per l'auricolare adatti agli spinotti Jack 3,5 mm, sara' necessario munirsi di un adattatore.

Dopo essersi accertati che il registratore e' in buone condizioni di funzionamento (testine pulite e, se possibile, smagnetizzate) si puo' procedere come segue:

. 1) registrare sul nastro un programma che si trovi in memoria;

. 2) leggere in memoria un programma che si trovi sul nastro.

PROVA 1 - Operare cosi':

. premere il tasto NEW e poi NEW LINE;

. scrivere 10 REM STO PROVANDO A REGISTRARE e poi NEW LINE;

. mettere il registratore in grado di registrare la voce con i collegamenti al calcolatore staccati;

. avviare il nastro per registrare;

. registrare parlando PROVA DI REGISTRAZIONE e fermare il nastro;

. inserire il collegamento MIC (o REC) tra calcolatore e registratore;

. riavviare il nastro;

. premere subito sulla tastiera SAVE e poi NEW LINE.

A questo punto si vede scomparire la scrittura dallo schermo, esso diventa grigio, poi si vedono comparire delle righe orizzontali ed alla fine ricompaiono le scritte di prima, attendere 10 secondi e fermare il registratore.

Il programma e' stato registrato sul nastro.

Se il registratore ha il controllo del livello di registrazione, bisogna assicurarsi tramite l'apposito

indicatore che il segnale sia registrato ad un livello sufficientemente alto.

Prima di fare la seconda prova si deve cancellare lo schermo e azzerare la memoria premendo NEW e poi NEW LINE; si vedra' ricomparire il K nel quadratino nero in fondo al video a sinistra. Riavvolgere il nastro al numero di giri prima della registrazione appena fatta.

PROVA 2 - Operare cosi':

. staccare i collegamenti registratore/calcolatore;
. cercare sul nastro la frase: PROVA DI REGISTRAZIONE, tenendo basso il volume;
. dopo la frase si sente un BRR... e poi silenzio; fermare il registratore appena inizia il silenzio;
. inserire il collegamento EAR (o MONITOR) tra registratore e calcolatore ed alzare il volume del registratore;
. riavviare il nastro e premere subito LOAD e poi NEW LINE;
. lo schermo diventa grigio e poi appare la lista del programma;
. fermare il registratore.

Se le due prove non hanno dato buon esito ritentare seguendo con precisione le istruzioni.
Alcuni utenti non sono riusciti facilmente ad ottenere la registrazione dei programmi ed il loro caricamento in memoria. In tutti questi casi o non venivano seguite puntualmente le istruzioni, o il registratore non era in buone condizioni, o i cavetti di collegamento si erano rovinati.

3.2. MONTAGGIO NUOVA ROM E MASCHERINA TASTIERA

L'operazione di sostituzione della ROM e' molto semplice. Per facilitarla ulteriormente si consiglia di acquistare un "estrattore" e un "inseritore" della "OKTOOL", reperibili presso tutte le Sedi G.B.C. rispettivamente con i numeri di codice: SM/5265-00 e SM/5280-00.

Schematizziamo la procedura:

. estrarre le 5 clips che tengono chiuso il contenitore di plastica del calcolatore;
. togliere il coperchio di plastica mettendo allo scoperto i diversi componenti del calcolatore;
. togliere la vecchia ROM, facilmente riconoscibile dalla scritta ROM, situata nell'angolo destro in alto:
- o con l'attrezzo estrattore mediante una leggera

trazione verso l'alto;
- o manualmente facendo leva con un piccolo caccia=
  vite tra la ROM e lo zoccolo sottostante;

. inserire la nuova ROM:
  - o con l'attrezzo inseritore, dopo avervi delica=
    tamente inserito la nuova ROM, appoggiandolo sul=
    lo zoccolo rispettando la posizione della tacca
    ed esercitando una leggera pressione;
  - o manualmente prendendo la ROM tra il pollice e
    l'indice ed inserendola nello zoccolo rispettando
    la posizione della tacca;

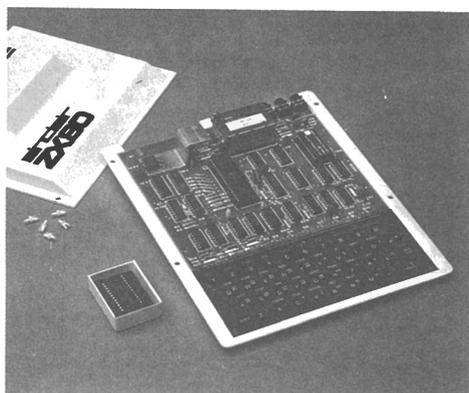



Fig. 3.2. Lo ZX80 aperto

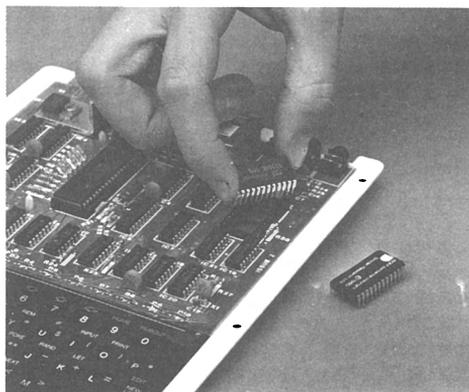



Fig. 3.3. La sostituzione della ROM

. togliere le 2 clips nere poste sul bordo inferiore della tastiera, applicare la nuova mascherina sopra la vecchia, facendo combaciare i fori per le clips, quindi rimettere le 2 clips nere;

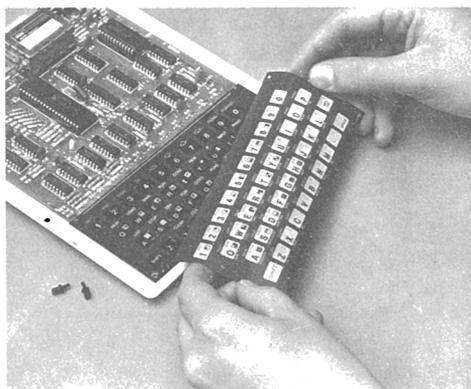



Fig. 3.4. Applicazione mascherina tastiera

. rimontare il coperchio di plastica e fissarlo con le 5 clips tolte inizialmente.

A questo punto voi non disponete piu' del primitivo ZX80, ma di un nuovo calcolatore, dotato di un nuovo Sistema Operativo e di un nuovo Basic.
Per quanto concerne le prove del nuovo calcolatore vale quanto si dice nel prossimo paragrafo per lo ZX81.

3.3. INSTALLAZIONE DELLO ZX81

Per l'installazione dello ZX81 vale tutto quello che si e' detto nel Paragrafo 3.1. a proposito dello ZX80, solo che si deve fare riferimento alla Fig. 3.5. per lo schema di collegamento, infatti nello ZX81 si trovano sul lato sinistro le prese che si trovano dietro nello ZX80. Rimane dietro, ma spostata verso destra la grande fessura che serve per collegare la RAM aggiuntiva.

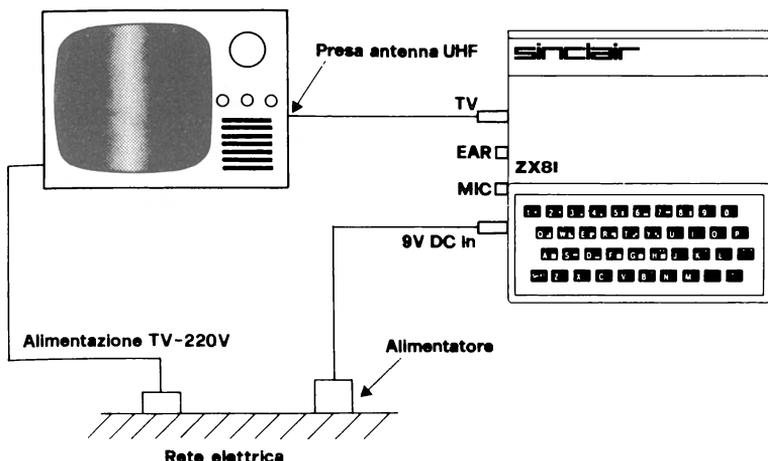



Fig. 3.5. Schema di collegamento per lo ZX81

Rimane del pari valido il programma per il controllo del calcolatore esposto nel paragrafo precedente, pur di leggere al posto dei 5 punti segnati con (* n) quello che segue:

.(* 1) - P Entra PRINT (parola scritta sopra il tasto P) seguita da uno spazio ed il cursore passa allo stato L.

.(* 2) - NEW LINE E' tutto uguale allo ZX80 salvo che i numeri appaiono per effetto della virgola solo in due colonne e che per avere la lista del programma si deve scrivere LIST e poi premere NEW LINE.

.(* 3) - SHIFT + 1 La spiegazione e' uguale a quella dello ZX80.

.(* 4) - SHIFT + B La spiegazione resta invariata.
.(* 5) - SHIFT + A Entra il comando NEW.

Per quanto riguarda invece la PROVA 1 e la PROVA 2 si hanno delle differenze e quindi vengono qui ripetute le due procedure per provare a registrare su nastro un programma e per provare a rileggerlo in memoria. La differenza fondamentale sta nel fatto che con lo ZX81 si deve memorizzare un programma su nastro assegnandogli un nome.

PROVA 1 - Operare cosi':

. premere il tasto NEW e poi NEW LINE;

. scrivere 10 REM STO PROVANDO A REGISTRARE e poi NEW LINE;

. inserire il collegamento MIC (o REC) tra calcolatore e registratore;

. avviare il nastro (se il nastro e' all'inizio far superare la zona dove non si puo' registrare);

. premere subito sulla tastiera SAVE "PROVA REG" e poi NEW LINE.

A questo punto si vede scomparire la scrittura dallo schermo, esso diventa grigio, poi si vedono comparire delle righe orizzontali ed alla fine compare 0/0 in basso a sinistra, attendere 10 secondi e fermare il registratore.

Il programma e' stato registrato sul nastro preceduto dal nome del programma, PROVA REG nel nostro caso.

Se il registratore ha il controllo del livello di registrazione assicurarsi, tramite l'apposito indicatore, che il segnale sia registrato ad un livello sufficientemente alto.

Prima di fare la seconda prova si deve cancellare lo schermo e azzerare la memoria premendo NEW e poi NEW LINE; si vedra' ricomparire il K nel quadratino nero in fondo al video a sinistra. Riavvolgere il nastro almeno fino al numero di giri prima della registrazione appena fatta.

PROVA 2 - Operare cosi':

. staccare i collegamenti registratore/calcolatore;

. inserire il collegamento EAR (o MONITOR) tra registratore e calcolatore e mettere alto il volume del registratore;

. avviare il nastro e premere subito LOAD "PROVA REG" e poi NEW LINE;

. lo schermo diventa grigio e poi compare 00;

. per ottenere la lista del programma premere NEW LINE;

. fermare il registratore.

Se le due prove non hanno dato buon esito ritentare seguendo con precisione le istruzioni.

Come avete potuto notare non e' piu' necessario registrare a voce il nome del programma, dato che il comando SAVE richiede anche il nome del programma. Su un nastro possono essere quindi memorizzati piu' programmi, ciascuno viene preceduto dal suo nome. Questo nome serve poi al comando LOAD per andare a ricercare sul nastro il programma desiderato. Il comando LOAD puo' essere usato anche scrivendo: LOAD "", dove "" e' la stringa nulla ottenuta premendo 2 volte il tasto P". In questo caso viene caricato il primo programma disponibile su nastro.

CAPITOLO 4

L A
P R O G R A M M A Z I O N E

4.1. IL PROGRAMMA

Un programma e' una serie ordinata di istruzioni il cui significato deve essere chiaro sia a chi le prepara, sia a chi le riceve. Nella vita comune si hanno molti esempi di programmi: una ricetta di cucina e' un programma, le istruzioni per far funzionare un qualsiasi apparecchio sono un programma, la lavabiancheria funziona seguendo un programma. Nel caso dei calcolatori, chi riceve le istruzioni e' una macchina predisposta a fare solo una serie ben definita di operazioni, niente di piu'. Solo che si ha la liberta' di impartire al calcolatore infinite sequenze delle istruzioni che esso puo' eseguire, combinandole in modi diversi; da questo dipende la grande versatilita' di queste macchine.

La sequenza delle istruzioni per il calcolatore deve essere preparata con cura, non si possono fare errori, esso infatti non possiede la fantasia ed il buon senso con cui un essere umano puo' interpretare delle istruzioni incomplete ricevute da un altro.

Il programma deve essere scritto in un linguaggio adatto al calcolatore e deve consentire di risolvere un determinato problema.

4.2. LO STUDIO DEL PROBLEMA

Prima di pensare alla stesura di un programma per il calcolatore, si deve esaminare il problema che si vuole risolvere, esponendolo in modo chiaro e completo. Devono essere descritti i dati iniziali sui quali si deve lavorare. Analogamente devono essere chiaramente descritti i risultati che si vogliono ottenere. Deve essere definita una procedura operativa che, utilizzando i dati iniziali, arrivi a produrre i dati finali. Di norma queste procedure operative prendono il nome di algoritmi. Tutti ricordano l'algoritmo (o formula) risolutivo delle equazioni di secondo grado. La

procedura operativa deve anche, in qualche modo, essere decritta con la maggior completezza possibile.
Quanto detto sopra risulta in generale abbastanza difficile per tutti, si tende sempre a dimenticare qualcosa.

Esaminiamo brevemente come si procede per risolvere manualmente un problema; vale in generale lo schema seguente:

. 1) si scrivono in una zona del foglio i dati iniziali;
. 2) si eseguono in sequenza delle operazioni aritmetiche;
. 3) a seconda dei risultati ottenuti si operano delle scelte sul tipo di operazioni con cui proseguire;
. 4) si ripetono un certo numero di volte dei gruppi di operazioni;
. 5) si scrivono in una zona del foglio i risultati ottenuti.

Per quanto riguarda il punto 3), e' chiaro che deve essere stata presa in precedenza una decisione su quale metodo di calcolo adottare.

4.3. IL PASSAGGIO DAL PROBLEMA AL PROGRAMMA

Tutti i linguaggi di programmazione mettono a disposizione del programmatore istruzioni adatte per svolgere le operazioni elencate nello schema esposto nel precedente paragrafo.

Per i principianti risulta abbastanza difficile passare dal problema al programma, anche se hanno studiato bene le possibilita' del linguaggio che vogliono adoperare. Si ha una specie di blocco mentale! Eppure non e' difficile. Si deve solo "rompere il ghiaccio" e cioe' cominciare a scrivere programmi. Si faranno molti errori, ma e' proprio facendo errori che si impara. La programmazione e' proprio una disciplina che deve essere studiata, ma che, soprattutto, deve essere praticata. Puo' essere molto utile leggere programmi scritti da altri e gia' funzionanti, purche' questi "altri" siano dei buoni programmatori, cioe' programmino in modo semplice e chiaro.

Una delle prime cose da imparare e' come riuscire a schematizzare il problema che si vuole risolvere. Non esistono metodi codificati ed obbligatori per raggiungere questo scopo. Negli ultimi anni sono state sviluppate delle metodologie che aiutano a ben programmare, appoggiandosi anche a linguaggi appositamente studiati per realizzarle.

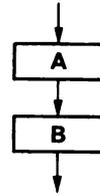
4.4. LE SITUAZIONI LOGICHE

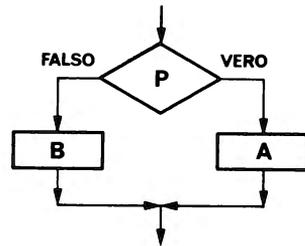
Durante lo studio di una qualunque procedura risolutiva si vede che emergono 3 possibili situazioni logiche, esse sono:

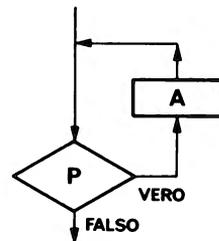
. a) sequenza;
. b) diramazione;
. c) iterazione.

Passiamo alla descrizione di queste situazioni servendoci sia della loro descrizione verbale che di un diagramma grafico che ben si presta a rappresentarle.

SEQUENZA

Dopo l'operazione A esegui l'operazione B

DIRAMAZIONE

Se la condizione P e' vera allora esegui l' operazione A, altrimenti (condizione P falsa) esegui l'operazio= ne B.

ITERAZIONE

Esegui l'operazione A fino a quando la condizione P rima= ne vera.

Si puo' riuscire a schematizzare ogni procedura operativa in una combinazione delle 3 situazioni logiche descritte. Queste 3 situazioni possono essere considerate le strutture base della programmazione. Esse hanno una caratteristica comune: un solo punto di entrata ed un solo punto di uscita.

4.5. STESURA DI DIAGRAMMI A BLOCCHI O DI SCHEMI DESCRITTIVI DEL PROGRAMMA

Riportiamo alcuni esempi di studio di problemi per poter arrivare alla stesura dei relativi programmi.

ESEMPIO 1

"Leggere un numero dall'esterno e stabilire se e' maggiore di 57."

Descrizione verbale:

. 1) leggere il numero N;
. 2) confrontare il numero N con 57; se N > 57 andare al punto 3), se no andare al punto 5);
. 3) scrivere: N > 57;
. 4) andare al punto 6);
. 5) scrivere: N < 57;
. 6) fine della sequenza.

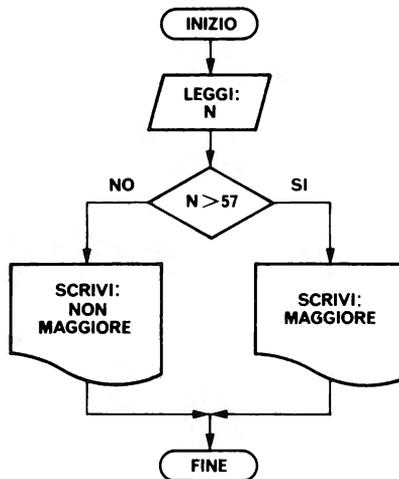



Fig. 4.1. Diagramma a blocchi

Sta al lettore decidere se ritiene per lui piu' chiara la descrizione verbale o il diagramma. Possiamo osservare che la procedura precedente puo' essere descritta inizialmente mediante la sequenza di 2 operazioni: A e B; dove A e' l'operazione di lettura di N e B e' l'operazione di analisi su N. Il blocchetto B si particolarizza poi in una struttura di diramazione. Se nel diagramma precedente si disegna un tratteggio che comprenda i blocchetti compresi tra "LEGGI:N" e "FINE", appare chiaramente quanto ora esposto.

ESEMPIO 2

"Leggere 3 numeri A, B, C e calcolare la media M dei 3 numeri."

Descrizione verbale:

. 1) leggere il numero A;
. 2) leggere il numero B;
. 3) leggere il numero C;
. 4) calcolare M = (A+B+C)/3
. 5) stampare la media M calcolata.

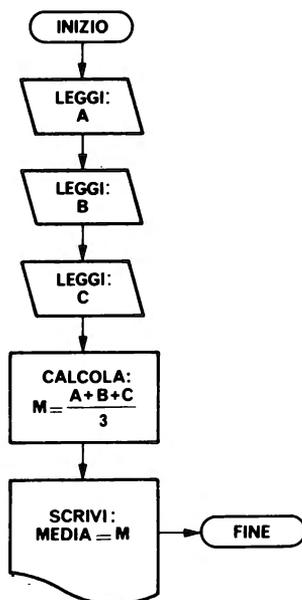



Fig. 4.2. Diagramma a blocchi media 3 numeri

Come si vede si tratta puramente di una procedura che richiede la struttura sequenziale applicata ripetutamente.

ESEMPIO 3

"Leggere 10 numeri e calcolare la media M dei 10 numeri letti."

Descrizione verbale:

. 1) porre a zero la somma S;
. 2) porre I=1 per contare la ripetizione della operazione di lettura;
. 3) confrontare I con 10; se I minore o uguale a 10 andare al punto 4), se no andare al punto 7);
. 4) leggere N;
. 5) aggiungere N alla somma S;
. 6) aggiungere 1 al contatore I, e tornare al punto 3);
. 7) calcolare M=S/10;
. 8) scrivere: MEDIA=M.

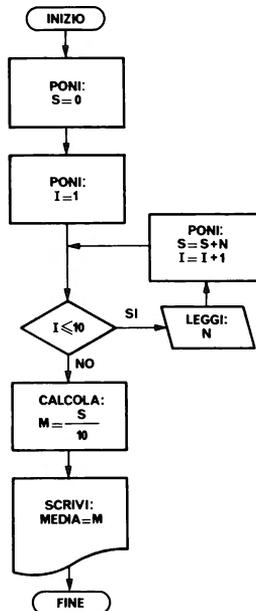
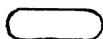
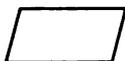



Fig. 4.3. Diagramma a blocchi media 10 numeri

Se osserviamo il diagramma di quest'ultimo esempio vediamo ordinatamente una struttura sequenziale, una struttura iterativa e poi ancora una struttura sequenziale collegate in sequenza tra loro.

Gli esempi riportati necessitano di qualche commento.

I diagrammi a blocchi sono degli schemi grafici dove compaiono dei disegni convenzionali con all'interno delle scritte esplicative; i diversi blocchetti sono collegati tra loro da segmenti orientati che danno il senso di percorrenza dello schema.
I simboli grafici usati sono:

per inizio e fine programma

per ingresso di dati

per uscita di messaggi

per confronto e scelta

per operazioni di calcolo

Esistono anche altri simboli, ma non e' il caso di indicarli tutti ora.

Nel descrivere i problemi, sia verbalmente che graficamente abbiamo usato dei nomi come: N, A, B, C, S, M. Questi nomi servono per indicare delle variabili, cioe' dei contenitori di dati. Sono state usate anche delle costanti numeriche come: 57, 3, 10.

Nel terzo esempio, dovendo fare una sommatoria S, si e' messa in evidenza l'operazione di azzeramento della variabile S prima di iniziare a sommare. Sempre nello stesso esempio per controllare l'esecuzione ripetitiva di lettura di un numero si e' fatto uso di un contatore I, posto inizialmente al valore 1 e incrementato di 1 ogni volta che si legge un numero. Tale contatore e' proprio quello che deve essere analizzato per poter uscire dal ciclo, e' la condizione che, fino a quando si mantiene vera (I<=10) fa eseguire l'iterazione.

Fino ad ora non si e' ancora parlato di stendere un programma, ma solo di studiare il problema. Per ogni problema e' stata fatta una analisi e si e' arrivati alla descrizione verbale o grafica della procedura risolutiva. Gli esempi scelti sono molto semplici e non si sono descritti completamente i dati di ingresso. In realta' si dovrebbe precisare la natura dei dati: numeri interi o decimali, numero di decimali desiderato nel risultato. Ogni problema dovrebbe essere accompagnato da una lista delle variabili in gioco, in modo che risulti chiaro per tutti cosa sono e a cosa servono. Nel caso dell'esempio 3:

Elenco variabili:
- I contatore per il controllo del ciclo di lettura puo' variare da 1 a 10;
- S variabile per calcolare la somma dei numeri letti;
- M variabile per calcolare la media con x decimali;
- N variabile per contenere il numero letto ogni volta tale variabile deve contenere un numero intero (o decimale).

4.6. LA PROVA DEL PROGRAMMA

Per ogni problema devono essere preparati dei dati di prova. Cioe', tenendo presente la natura del problema si devono preparare e calcolare tutti i casi limite con i quali e' necessario provare il programma. E' inutile, per esempio,

provare un programma di ordinamento solo con dei dati gia' in ordine!

La prova del programma e' un momento molto delicato. In generale un programma non e' mai giusto a priori. Gli errori possono dipendere da diverse cause.

Gli errori piu' semplici sono quelli inerenti alla grammatica e sintassi del linguaggio usato. Oltretutto i sistemi operativi o i traduttori dei linguaggi aiutano nel trovare molti di questi errori.

Gli errori possono essere dovuti ad una cattiva codifica del problema, tipo un richiamo sbagliato, e questo tipo di errore non puo' essere segnalato dal sistema a meno che il punto richiamato non esista.

Ci sono inoltre, e sono i piu' gravi, gli errori di logica nell'analisi del problema. Se il programma in prova, non arriva alla sua fine logica o se i risultati non sono quelli attesi, si deve riprendere con pazienza in esame tutto dall'inizio.

4.7. LA DOCUMENTAZIONE DEL PROGRAMMA

Quando il problema e' stato sufficientemente approfondito ed e' disponibile:

. l'analisi del problema;
. la descrizione verbale della procedura o il diagramma a blocchi;
. l'elenco delle variabili;

si puo' passare alla codifica del programma in un linguaggio adatto per il calcolatore sul quale il programma deve essere provato. La codifica viene portata avanti seguendo la descrizione verbale della procedura o il diagramma a blocchi ed avendo a disposizione tutte le note relative alle variabili da usare ed alla procedura.

L'analisi del problema puo' essere portata avanti in due modi:

. a) tenendo presente solo la logica del problema;
. b) tenendo presente sia la logica del problema che la natura del linguaggio che si usera' per la codifica del programma.

Nel caso b) sara' piu' semplice la fase di codifica, mentre nel caso a) si dovra' adattare la logica del problema alle esigenze del linguaggio.

Dopo la prova definitiva del programma, cioe' quando tutto funziona, si devono preparare le norme operative per l'uso del programma.

Ogni programma per il calcolatore deve essere documentato, cioe' deve esistere:

- il testo del problema;
- l'analisi del problema;
- il diagramma a blocchi o la descrizione verbale;
- l'elenco di tutte le variabili usate;
- l'elenco dei casi prova significativi;
- la lista del programma;
- le norme operative per l'uso del programma.

E' chiaro che in questo libro si fanno degli esempi abbastanza semplici, per i quali la documentazione e' necessariamente limitata.

4.8. I DATI E LA LORO ORGANIZZAZIONE

I dati sui quali operano i programmi possono essere:

- dati singoli;
- dati organizzati a gruppi.

Sulla prima categoria di dati non c'e' molto da dire a parte che i dati possono essere di tipo diverso, cioe' numerici interi, numerici decimali, alfanumerici (stringhe per il Basic).

Per quanto riguarda la seconda, si possono avere gruppi di dati tutti dello stesso tipo, per esempio 100 numeri da ordinare, oppure gruppi di dati di tipo diverso, ma legati insieme da una qualche caratteristica, per esempio notizie riguardanti tutte la stessa persona. Questi ultimi si sogliono raggruppare con il nome di record. Il record anagrafico di una persona comprende diversi elementi, per esempio:

- nome e cognome;
- indirizzo;
- citta';
- numero di telefono;
- CAP;
- data di nascita;
- luogo di nascita;
- ecc.;

e questi dati non sono necessariamente tutti dello stesso tipo.

E' abbastanza raro avere a che fare con un solo record di dati; di solito si hanno tanti record di dati e questi costituiscono un archivio. Nel linguaggio della programmazione gli archivi si chiamano file o flussi.

I file o archivi o flussi sono insiemi di dati registrati su un supporto. I supporti piu' comuni sono: fogli di carta, dischi magnetici, nastri magnetici, bande di carta, schede perforate. In dipendenza dalla natura intrinseca del supporto il file puo' essere solo di OUTPUT (uscita) o anche di INPUT (entrata). Ogni insieme e' costituito da un certo numero di elementi; ogni elemento prende il nome di record logico. Ogni record logico e' costituito da piu' dati elementari che prendono il nome di campi (field). Il record logico e' un insieme di campi che e' opportuno o registrare uno dopo l'altro o registrare in modo tale che tutti i campi siano accessibili contemporaneamente o globalmente in fase di elaborazione.

L'aspetto logico dei file e' quello che interessa il programma elaborativo; esiste pero' anche un secondo aspetto ed e' quello fisico. Cioe' come fisicamente si registrano i record sui supporti. Ogni supporto consente di registrare con determinate caratteristiche, e non e' il caso di sviluppare qui questo argomento.

Nel caso dei calcolatori Sinclair i dati possono essere organizzati in record e quindi simulare l'organizzazione dei file, ma il tutto deve essere registrato nella memoria del calcolatore e spostarsi su nastro magnetico insieme al programma. Cioe' non si ha la possibilita' di chiamare da programma e leggere o scrivere un file su nastro; si deve lavorare nella memoria e poi memorizzare alla fine della elaborazione di nuovo tutto su nastro. Nel Capitolo 9 si dedica un paragrafo a questo argomento.

CAPITOLO 5

I L L I N G U A G G I O B A S I C

5.1. CARATTERISTICHE DEL LINGUAGGIO

Il BASIC e' un linguaggio simbolico ad alto livello di tipo interpretativo. Questo significa:

. il programmatore scrive le istruzioni usando delle parole simboliche e dei simboli abbastanza vicini al normale linguaggio (in inglese) la cui logica e' piu' orientata alla risoluzione dei problemi che non al funzionamento del calcolatore;
. ogni istruzione del linguaggio corrisponde ad un gruppo di istruzioni in linguaggio macchina;
. la traduzione da linguaggio simbolico a linguaggio macchina avviene contemporaneamente alla esecuzione del programma, cioe' ogni frase viene prima interpretata e poi eseguita.

In molti altri linguaggi simbolici di programmazione la fase di traduzione e' completamente separata dalla fase di esecuzione e la precede. Questi tipi di linguaggi si definiscono compilativi (quelli ad alto livello tipo FORTRAN, COBOL e altri) o assemblativi (quelli a basso livello tipo ASSEMBLER). Questo fa si che, quando si scoprono degli errori in fase esecutiva, risulta abbastanza macchinoso correggerli. Infatti si deve correggere il programma simbolico, rifare la traduzione e rifare la preparazione finale del programma da provare.

Con il Basic, invece, se si scoprono degli errori in fase esecutiva, si correggono le frasi sbagliate, gia' presenti in memoria, e si riprova. La messa a punto di un programma risulta piu' veloce.

Il BASIC ha sicuramente lo svantaggio di essere un po' piu' lento in fase esecutiva rispetto ai linguaggi compilativi o assemblativi, ma la maneggevolezza dei programmi e' tale che vale la pena di usarlo.

Si deve fare attenzione e ricordarsi se un programma in prova e' stato modificato rispetto alla versione gia'

memorizzata, per esempio su nastro, e quindi memorizzare nuovamente la versione aggiornata.

In questo manuale non si pretende di fare un trattato sul BASIC standard, ma solo di insegnare le caratteristiche delle due implementazioni del linguaggio disponibili sui calcolatori SINCLAIR ZX81, ZX80-NUOVA ROM e ZX80.

5.2. COME SI SCRIVONO I PROGRAMMI

Il programma e' formato da linee numerate da 1 a 9999 al massimo; ogni linea contiene una istruzione. La numerazione progressiva delle linee rappresenta anche l'ordine di esecuzione delle istruzioni del programma. Si usa numerare le istruzioni del programma con numeri non consecutivi, in tale modo e' possibile fare delle inserzioni di linee senza dovere rinumerare le altre. Si possono usare i numeri: 10, 20, 30, ecc..

Le istruzioni sono formate dalle PAROLE CHIAVE proprie del linguaggio e dalle PAROLE SIMBOLICHE inventate dal programmatore per indicare gli OPERANDI. Esistono delle regole per la formazione delle parole simboliche.

Il programma si scrive seguendo la nascita delle istruzioni nella parte bassa del video; il sistema segnala eventuali errori di scrittura. Le correzioni si apportano muovendosi lungo la linea per mezzo dei tasti che spostano il cursore a destra e a sinistra oppure usando il tasto RUBOUT per cancellare. Quando una istruzione e' completa e non contiene errori la pressione del tasto NEW LINE la fa accettare dal sistema; questo significa che l'istruzione viene memorizzata ed appare nella parte alta del video al posto giusto in base al numero di linea crescente. Se si riscrive una istruzione che esiste gia', la nuova istruzione va a sostituire la vecchia, cioe' quella che aveva lo stesso numero di linea. Per modificare una istruzione gia' sistemata nella parte alta dello schermo si procede cosi':

. si sposta il puntatore di linea alla istruzione in questione;
. si usa il tasto EDIT per far apparire la stessa istruzione nella parte bassa dello schermo;
. si corregge la linea e poi si preme NEW LINE.

La nuova istruzione va a sostituire la vecchia.

5.3. I DUE MODI DI FUNZIONAMENTO

Il calcolatore puo' funzionare in:

- . MODO IMMEDIATO;
- . MODO DIFFERITO.

Modo immediato significa che quando si preme NEW LINE per fare accettare una istruzione questa viene immediatamente eseguita e non rimane memorizzata dopo l'esecuzione. Per ottenere questo modo di funzionamento si deve scrivere l'istruzione senza farla precedere dal numero di linea.

Modo differito significa che le istruzioni vengono memorizzate e per farle eseguire si deve dare il comando che manda in esecuzione il programma.

Durante l'esecuzione del programma si possono avere degli arresti programmati nell'esecuzione ed intervenire in modo immediato per analizzare risultati parziali senza disturbare il programma presente in memoria.

Durante l'esposizione delle istruzioni del linguaggio viene detto quando una istruzione non e' eseguibile in uno dei due modi.
Inoltre, per ogni istruzione, qualora non sia valida su uno dei modelli del SINCLAIR, viene indicato per quale calcolatore e' utilizzabile.

5.4. CATEGORIE DI ISTRUZIONI

Le istruzioni del linguaggio possono essere divise in tre gruppi:

- . istruzioni di tipo dichiarativo;
- . istruzioni di tipo esecutivo;
- . comandi di sistema.

Al primo gruppo appartengono le istruzioni che aiutano l'interprete Basic a lavorare. Al secondo gruppo appartengono le istruzioni che vengono effettivamente svolte per eseguire un programma. Al terzo gruppo appartengono le istruzioni che interagiscono fra programma e sistema.

Le istruzioni esecutive si possono classificare in diversi gruppi:

- . istruzioni di assegnazione;
- . istruzioni di controllo;

. istruzioni di ingresso e uscita dei dati;
. istruzioni funzionali (funzioni);
. istruzioni varie e di servizio.

5.5. I COMANDI DI SISTEMA

Le istruzioni di questo gruppo sono comandi di sistema e sono eseguiti prevalentemente in modo immediato. Per ogni comando viene specificato se si puo' usare anche in modo differito e quale effetto produce.

NEW

azzera la memoria del calcolatore, deve essere usato prima di iniziare a scrivere un nuovo programma. Non ha senso usarla in un programma.

RUN

azzera tutte le variabili del programma presente in memoria e ne fa partire l'esecuzione dalla linea con il numeero di linea minore. Non ha in generale senso usarla in un programma.
Si puo' anche scrivere: RUN numero-linea, in questo caso, dopo l'azzeramento delle variabili, l'esecuzione parte dal numero-linea dato.

LIST

lista sul video il programma presente in memoria. Se esso supera le 22 linee, appaiono solo le prime 22. Lo schermo puo' contenere 24 linee, ma dopo la lista viene mantenuta una linea in bianco e l'ultima linea e' del cursore. Per listare la parte restante del programma si puo' scrivere: LIST n. Nello ZX80 si ottiene:
- se la n e' gia' sullo schermo, compare il puntatore di linea a marcarla;
- se essa non e' gia' sullo schermo ed n riferisce una delle 2 linee seguenti lo schermo scorre ed aggiunge la o le due linee, con il puntatore alla linea n;
- se n indica una linea oltre l'ultima presente + 2, compare il programma dalla linea precedente la n in avanti, sempre con il puntatore alla linea n.
Nello ZX81 e nello ZX80-Nuova ROM qualunque parte del programma sia presente sul video, se si scrive LIST n, appare la

parte di programma che inizia alla linea n.
Non ha senso usare LIST in un programma, il programma si interrompe e lista se stesso.

LOAD

valido solo per ZX80. Trasferisce un programma in memoria dal nastro esattamente nello stato in cui era quando e' stato memorizzato, quindi possono esserci dei contenuti nelle variabili. La memoria viene azzerata prima del caricamento del programma, come se si fosse usato il comando NEW.

LOAD""
LOAD"nome-pr"

valido per ZX81 e ZX80-nuova ROM. Nella forma LOAD"" carica da nastro il primo programma che trova ("" e' la stringa nulla ottenuta premendo 2 volte il tasto P" e non il tasto "" che serve per introdurre il carattere apice in una stringa). Nell' altra forma carica da nastro il programma di nome "nome-pr" facendo scorrere eventuali altri programmi presenti. Il programma viene caricato nello stato in cui si trovava al momento della memorizzazione, quindi anche con variabili non vuote.

NOTA: Se si usa la LOAD per caricare un programma insieme ai contenuti delle variabili, non si puo' usare il comando RUN per mandarlo in esecuzione. Infatti RUN azzera i contenuti delle variabili. In questi casi si deve usare in modo immediato il comando: GOTO n, dove n e' il numero di linea della prima linea del programma. L'istruzione LOAD puo' essere usata anche da programma, solo che essa interrompe il programma in esecuzione e fa caricare il nuovo programma.

SAVE

valido per lo ZX80. Memorizza su nastro il programma che si trova in memoria insieme ai contenuti delle variabili.

SAVE"nome-pr"

valido per ZX81 e ZX80-Nuova ROM.

Memorizza su nastro il programma che si trova in memoria insieme ai contenuti delle variabili assegnandogli il nome "nome-pr". Non e' accettata la stringa nulla al posto del nome del programma.

NOTA: Se si vuole essere sicuri che un programma memorizzi se stesso insieme alle sue variabili, si deve procedere cosi. Programmare alla fine logica della esecuzione del programma uno STOP preceduto da un messaggio esplicativo al video che chieda di fare le operazioni manuali inerenti al montaggio del nastro e poi di dare il comando CONT (CONTINUE). Dopo l'istruzione di STOP ci deve essere l'istruzione di SAVE. In tale modo il programma salva se stesso prima di terminare. Da quanto detto risulta che questa istruzione puo' essere usata anche da programma.

CONT

fa proseguire l'esecuzione di un programma dopo una fermata . Se il codice di errore e' 9, cioe' e' stata eseguita una istruzione STOP, il programma prosegue dalla linea seguente. Se il codice di errore non e' 9 o 0 il programma prosegue rieseguendo l'ultima istruzione. Se il codice di errore e' 0 allora CONT fa proseguire dal numero di linea del precedente messaggio di errore. Non ha senso usare questa istruzione in un programma.

5.6. TRATTAMENTO DEI DATI NELLO ZX80

I dati possono essere COSTANTI e VARIABILI. Le costanti sono introdotte direttamente nelle linee di programma senza ricevere l'assegnazione di un nome simbolico. Le variabili invece sono individuate da un nome simbolico. Le caratteristiche di ogni possibile tipo di dato sono identiche per le costanti e per i contenuti delle variabili.

Le COSTANTI possono essere:

. NUMERI INTERI, con segno, compresi tra -32768 e +32767;
. STRINGHE DI CARATTERI alfanumerici delimitate dagli apici, che ovviamente non possono far parte della stringa.

Es.: "OGGI PIOVE". Le stringhe possono essere lunghe a piacere. Una stringa senza caratteri ("") viene chiamata stringa-nulla e si ottiene premendo 2 volte il tasto Y".

Analogamente le VARIABILI possono essere:

. VARIABILI NUMERICHE INTERE;
. VARIABILI STRINGA ALFANUMERICHE.

Le variabili numeriche intere hanno nomi simbolici che devono sempre iniziare con una lettera e possono contenere, dopo il primo carattere, sia cifre che lettere e possono essere lunghi a piacere. Naturalmente i nomi lunghi occupano piu' spazio in memoria di quelli corti e quindi e' meglio limitare il numero dei caratteri dei nomi delle variabili. Le variabili numeriche possono contenere numeri compresi tra -32768 e +32767. Esse sono memorizzate in due byte, nel primo si trovano le cifre meno significative e nel secondo le piu' significative. I numeri negativi sono memorizzati nella forma di complemento a due con il primo bit del byte alto a 1. I numeri positivi hanno sempre il primo bit del byte alto a 0. Sono nomi validi: PAGA, I, A1, A2, A3, SCONTO, ecc..

Le variabili stringa hanno nomi simbolici formati da una lettera seguita dal carattere $ (dollaro) e possono contenere stringhe di qualsiasi numero di caratteri compatibilmente con la capacita' della memoria. Dato che le lettere dell'alfabeto sono 26, in un programma si possono avere al massimo 26 stringhe. Sono nomi validi: A$, G$,ecc..

Si possono avere VARIABILI NUMERICHE CON INDICE, cioe' gruppi di variabili, rappresentate globalmente dallo stesso nome, e distinte tra loro da un indice. In questo caso il nome delle variabili puo' essere formato da una sola lettera. Es.: A(I), dove A e' il nome del gruppo di variabili ed I e' la variabile intera che funge da indice nel gruppo. Il numero degli elementi non puo' superare 256. L'indice ha come primo valore 0.

Le variabili numeriche intere che vengono usate per controllare i cicli di programma (vedi istruzioni FOR/NEXT) sono chiamate VARIABILI DI CONTROLLO e possono avere il nome formato da una sola lettera.

Le variabili in programmazione sono da intendersi come "contenitori di dati" e quindi per le variabili hanno senso operazioni come: I=I+1, che dal punto di vista della matematica non hanno senso.
Le variabili singole in Basic vengono definite quando viene loro assegnato un valore iniziale, cioe' la prima volta che compaiono a sinistra di un = o in una istruzione

INPUT. Se in un programma si scrive I=I+1 senza avere prima scritto, per esempio I=0, si ha una segnalazione di errore; infatti I compare a destra di un = prima di essere stata definita. Le variabili con indice vengono invece definite usando una frase Basic di definizione, la DIM, istruzione di tipo dichiarativo.

Nel Capitolo 7 si mostra come i dati vengono memorizzati nella memoria del calcolatore.

5.7. TRATTAMENTO DEI DATI NELLO ZX81 E NELLO ZX80-NUOVA ROM

La nuova ROM consente di usare NUMERI INTERI o DECIMALI aventi almeno 9 cifre di precisione. Si arriva alle 10 cifre se i numeri si mantengono inferiori a 4294967296.

Il calcolatore accetta dalla tastiera numeri scritti in 3 modi:

. 1) NUMERI INTERI;
. 2) NUMERI CON IL PUNTO DECIMALE;
. 3) NUMERI IN NOTAZIONE ESPONENZIALE.

Per quanto riguarda i punti 1) e 2) non e' necessario fornire spiegazioni, basta solo ricordare che, usando i calcolatori elettronici, il punto decimale sostituisce la virgola decimale.

Il punto 3) si riferisce ai numeri scritti sotto forma di prodotto di un numero per una opportuna potenza di 10. Esempio:

5.27 = 527*10**-2 = 0.527*10**1 = 52.7*10**-1

E' chiaro che l'esempio potrebbe essere modificato all'infinito e questo non avrebbe molto senso (ricordiamo che "* significa moltiplicato" e che "** significa elevato alla potenza di"). E' invece interessante notare che ogni numero puo' essere scritto in forma esponenziale in modo univoco se si pongono tutte le cifre significative a destra del punto decimale, cioe' "0.cifre" e si usa un opportuno esponente per il moltiplicatore 10. Questo modo di scrivere i numeri viene chiamato "forma esponenziale normalizzata".

Nella forma esponenziale normalizzata vengono conservate tutte le cifre a partire dalla prima cifra significativa (diversa da zero) e questo consente, usando un numero prefissato di cifre, di conservare sia i numeri molto grandi che i numeri molto piccoli con una precisione predeterminata. L'esponente serve poi a dare la grandezza

reale del numero.
Inoltre non e' necessario conservare "0.", ma basta conservare le cifre dopo il punto; esse prendono il nome di "mantissa". Analogamente non e' necessario conservare "*10**", ma basta conservare l'esponente; esso prende il nome di caratteristica.

Per fare accettare dal calcolatore un numero in notazione esponenziale esso deve essere scritto cosi':

numeroExxx

dove: numero e' il numero scritto come intero o come deci=male e non necessariamente in forma normaliz=zata
E corrisponde convenzionalmente a "*10**";
xxx sta per un numero al massimo di 2 cifre con o senza segno e rappresenta l'esponente di 10.

Esempi:
0.527E1 che corrisponde a 5.27
527E-2 " " " 5.27
4.1E10 " " " 41000000000

Qualunque numero, non importa in quale modo sia stato immesso nel calcolatore, viene memorizzato in forma esponenziale normalizzata. Il sistema usa 5 byte per memorizzare un numero:

. 1 byte serve per la caratteristica;
. 4 byte servono per la mantissa.

Ovviamente, dato che il sistema conserva i numeri in forma binaria, la caratteristica rappresenta l'esponente da dare al moltiplicatore 2 (e non 10) per ottenere il numero, rappresentato a sua volta da una mantissa binaria.

Il byte della caratteristica (il cui valore puo' andare da 0 a 255) viene usato come esponente dopo avergli sottratto 128; in tale modo gli esponenti positivi variano apparentemente da 129 a 255 e realmente da 1 a 127, mentre quelli negativi variano apparentemente da 0 a 127 e realmente da -128 a -1. L'esponente reale 0 corrisponde all'esponente apparente 128.
Questo significa che la caratteristica dei numeri trattati varia in decimale da -39 a +38.

I 4 byte della mantissa servono per conservare una mantissa normalizzata usando le seguenti convenzioni:
. il primo bit del byte piu' alto serve per il segno: 0 per numeri positivi e 1 per numeri negativi;
. i 31 bit rimanenti servono per la mantissa, ma dato

che, essendo essa normalizzata comincia sicuramente con un bit 1, questo primo bit viene omesso, si ha cosi' un bit in piu' di precisione. Naturalmente nei calcoli viene tenuto conto anche del primo bit.
Il piu' grande numero memorizzabile in 32 bit e': 2**32-1, che e' appunto il numero 4294967295. Questo e' di 10 cifre, ma non e' il massimo numero di 10 cifre disponibile (sarebbero 10 cifre 9 consecutive) e quindi si dice che la precisione e' tra le 9 e le 10 cifre.
I numeri negativi non sono nella forma di complemento a 2, ma il primo bit e'uguale a 1 per indicare il se= gno meno, ed e' seguito dal valore assoluto del nume= ro privato del primo bit.

Il numero zero e' rappresentato dai 5 byte tutti al valore 0.

Quando i numeri vengono stampati si vedono solo 8 cifre significative eventualmente seguite da zeri e con l'ultima cifra significativa arrotondata, pero' il numero viene conservato in memoria con la precisione su esposta. Si possono provare queste caratteristiche del calcolatore scrivendo semplici esempi con calcoli di numeri molto grandi e molto piccoli.

Il nuovo BASIC del Sinclair consente quindi di trattare numeri interi o decimali senza le usuali distinzioni tra interi e decimali presenti nei Basic standard, dove l'aggiunta di un suffisso al nome della variabile o alla costante crea una distinzione tra interi e decimali.

Per quanto riguarda la formazione dei nomi delle variabili sono ancora valide tutte le regole esposte nel precedente paragrafo per lo ZX80.

Le VARIABILI NUMERICHE CON INDICE consentono l'uso di dimensioni multiple, cioe' non si ha piu' un solo indice, ma quanti indici si vuole. Il nome delle variabili numeriche con indice deve essere formato da una sola lettera. Gli indici partono dal valore 1. Per dimensionare le variabili con indice si usa la frase DIM, di tipo dichiarativo. Il numero degli elementi non ha, praticamente, limite.

Si puo' usare lo stesso nome per una variabile numerica semplice e per una variabile con indice ed il sistema le considera diverse.

Esempio:

```
10 REM MATRICE DI 3 RIGHE E 4 COLONNE
15 DIM M(3,4)
20 FOR I = 1 TO 3
30 FOR K = 1 TO 4
40 LET M(I,K) = I*10+K
```

```
45 PRINT M(I,K);"  ";
50 NEXT K
55 PRINT
60 NEXT I
```

Le VARIABILI DI CONTROLLO dei cicli hanno il nome formato da una sola lettera , ma sono anche esse memorizzate come numeri in forma esponenziale. Nel nuovo Basic sono spariti i numeri interi, cioe' i numeri contenuti in 2 byte.

Le regole per la formazione dei nomi delle STRINGHE sono invariate (lettera seguita da $), ma sono disponibili le VARIABILI STRINGA CON INDICE a dimensioni multiple, con la limitazione che tutti gli elementi stringa devono avere la stessa lunghezza. Nel dimensionamento di una matrice (variabile con indice di dimensioni multiple) di stringhe dopo le dimensioni deve essere presente un numero che definisce il numero dei caratteri di ogni elemento. Il numero delle dimensioni e', praticamente, a piacere. Esempio:

10 DIM A$(10,7) definisce una matrice di stringhe ad una dimensione (vettore), formata da 10 elementi stringa di 7 caratteri ciascuno.

100 DIM B$(7,5,10) definisce una matrice di stringhe a due dimensioni, formata da 7 righe e 5 colonne, quindi 35 stringhe, ognuna di 10 caratteri.

Non si puo' usare lo stesso nome per una variabile stringa semplice e per una variabile stringa con indice.
Per fissare una stringa in memoria si puo' definirla con la frase DIM senza assegnarle indici, ma solo la lunghezza. Esempio:

```
10 DIM A$(3)
```

definisce una stringa unica lunga 3 caratteri.

Gli indici possono essere: costanti, variabili, espressioni numeriche; essi vengono arrotondati all'intero piu' vicino; essi partono dal valore 1.

Ricordiamo che le variabili semplici devono essere inizializzate per cominciare ad esistere in un programma, cioe' ci deve essere o una frase LET di assegnazione o la variabile deve comparire dopo la parola INPUT e quindi ricevere un dato al momento dell'esecuzione del programma.
Le variabili con indice invece cominciano ad esistere al momento della DIM e vengono inizializzate o a zero o con spazi.

5.8. CARATTERI, OPERATORI E ESPRESSIONI

I caratteri del sistema sono i caratteri ASCII riportati nella Appendice A.

Gli OPERATORI ARITMETICI disponibili sono:

- . elevamento a potenza (**);
- . negazione unitaria (-);
- . moltiplicazione (*);
- . divisione (/);
- . addizione (+);
- . sottrazione (-).

Ad ogni operatore e' stata assegnata una priorita' operativa codificandola con un numero. Nella valutazione delle espressioni vengono eseguite prima le operazioni con priorita' piu' alta e l'espressione viene analizzata da sinistra a destra. Nelle espressioni si possono usare le parentesi e le operazioni contenute nelle parentesi hanno la precedenza rispetto alle altre. Tutti gli operatori aritmetici lavorano su variabili e costanti numeriche. L'operatore "+" puo' essere usato per concatenare tra loro due stringhe e questo equivale a scriverle una di seguito all'altra, pero' solo nello ZX81 e nello ZX80-Nuova ROM.

Gli OPERATORI RELAZIONALI disponibili sono:

- . uguale (=);
- . maggiore (>);
- . minore (<).

Questi operatori possono essere usati sia in espressioni aritmetiche che in espressioni che operano su stringhe.
Nello ZX81 e nello ZX80-Nuova ROM sono disponibili anche gli operatori relazionali che seguono:

- . minore uguale (<=);
- . maggiore uguale (>=);
- . diverso (<>).

Con gli operatori relazionali si possono formare espressioni condizionali e puo' essere analizzato il verificarsi o meno della condizione:

- . condizione vera;
- . condizione falsa.

Nel calcolatore ZX80 la condizione di verita' corrisponde al valore numerico -1, e la condizione di falsita' al valore numerico 0.
Nello ZX81 e ZX80-Nuova ROM la condizione di verita'

corrisponde al valore numerico 1 o meglio diverso da 0; la condizione di falsita' al valore numerico 0.

Una espressione condizionale fa nascere una variabile di tipo logico e questa variabile assume i valori sopra indicati.

Gli OPERATORI LOGICI disponibili sono:

- . NOT (negazione);
- . AND (una e l'altra);
- . OR (una o l'altra).

Anche gli operatori logici servono per costruire espressioni condizionali. Le espressioni condizionali possono far parte di espressioni e ad esse viene sostituito il valore della corrispondente variabile logica.

Seguono le tabelle delle priorita' degli operatori per le due versioni del Basic.

Priorita' Op. ZX80

Operazione	Priorita'
Funzioni	11
**	10
-(unitario)	9
*	8
/	7
+	6
-	6
=	5
>	5
<	5
NOT	4
AND	3
OR	2

Priorita' Op. ZX81/ZX80-Nuova ROM

Operazione	Priorita'
Indici	12
Slicing	12
Funzioni	11
**	10
-(unitario)	9
*	8
/	8
+	6
-	6
=	5
>	5
<	5
<=	5
>=	5
<>	5
NOT	4
AND	3
OR	2

Dal fatto che, nel primo caso le operazioni di moltiplicazione e divisione hanno diversa priorita' e nel secondo uguale priorita', deriva che calcolando la seguente espressione:

```
300/5*2
```

si abbia come risultato rispettivamente:

. nel primo caso 30 (ZX80), infatti viene prima eseguito 5*2 (priorita' 8) e poi diviso 300 per il risultato (priorita' 7);
. nel secondo caso 120 (ZX81 e ZX80-Nuova ROM), infatti essendo la priorita' delle due operazioni la stessa viene eseguita prima l'operazione piu' a sinistra 300/5 e poi moltiplicato il risultato per 2.

Nella tabelle compaiono delle operazioni delle quali non si e' ancora parlato e che verranno esposte piu' avanti.

Nello ZX80 si supplisce alla mancanza dell'operatore <> (diverso) usando l'operatore NOT; scrivendo per esempio NOT A = B, si indica proprio la condizione A diverso da B.

Gli operatori logici, nello ZX80, lavorano come di norma in molti altri Basic , cioe' vengono applicate le seguenti regole:

. Da X=2 AND Y=3	segue variabile logica=-1 se X=2 e Y=3; variabile logica=0 negli altri casi.
. Da X=2 OR Y=3	segue variabile logica=0 se X non = 2 e Y non = 3; variabile logica=-1 negli altri casi.
. Da Y= NOT X	segue variabile logica=-1 se X e Y sono diversi; variabile logica=0 se X=Y.
. Da Y=X AND Z	segue, se X=7 (in binario 00000111) e Z=117 (in binario 01100101), Y=5 (in binario 00000101).
. Da Y=X OR Z	segue per gli stessi valori di X e Z, Y=119 (in binario 01100111)

Nello ZX81 e nello ZX80-Nuova ROM valgono, invece le regole seguenti:

. Da X=2 AND Y=3	segue variabile logica=1 se X=2 e Y=3; variabile logica=0 negli altri casi.
. Da X=2 OR Y=3	segue variabile logica=0 se X <> 2 e Y <> 3; variabile logica=1 negli altri casi.
. Da Y= NOT X	segue Y=0 se X <> 0; Y=1 se X = 0.
. Da Y=X AND Z	segue Y=X se Z <> 0; Y=0 se Z = 0.
. Da Y=X OR Z	segue Y=X se Z = 0; Y=1 se Z <> 0.

Come si puo' osservare si hanno differenze di comportamento dovute alle differenze nella implementazione

dei due Basic.

5.9. ISTRUZIONE DI ASSEGNAZIONE

E' una istruzione di tipo esecutivo ed e' l'unica istruzione che consente di fare dei calcoli. Si chiama di assegnazione perche' viene "assegnato" un valore alla variabile che compare a sinistra dell' operatore =.
La forma e':

```
LET variabile = espressione
```

dove "variabile" e' il nome di una variabile e "espressione" e' una espressione formata usando nomi di variabili, costanti e operatori consentiti dal linguaggio.
La forma piu' semplice di questa istruzione e':

```
LET variabile = costante
```

e serve per assegnare un valore iniziale alle variabili. LET e' la parola chiave che definisce il tipo di istruzione.

5.10. ISTRUZIONI DI CONTROLLO

Fanno parte di questo gruppo le istruzioni che permettono di uscire dalla situazione logica di svolgimento sequenziale, per numero di linea crescente, del programma.

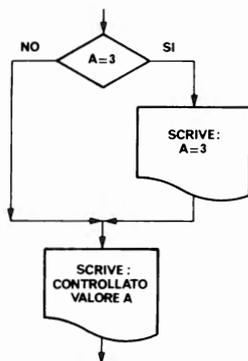
L'istruzione che realizza la condizione logica di diramazione e':

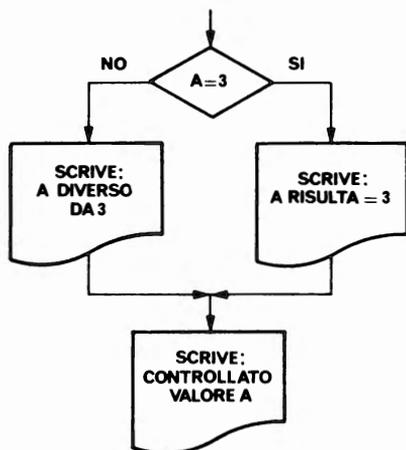
```
IF condizione THEN istruzione
```

dove "condizione" e' una espressione relazionale, della quale possono far parte anche operatori logici; la condizione viene analizzata e se essa risulta VERA viene eseguita "istruzione" dopo il THEN. Se la condizione analizzata risulta FALSA il programma prosegue dalla linea successiva. Dopo il THEN puo' essere presente una sola istruzione; essa puo' anche essere una istruzione di salto incondizionato ad un altro punto del programma. In questo caso l'istruzione IF/THEN prende anche il nome di salto condizionato.
Non ha senso usare questa istruzione in modo immediato.

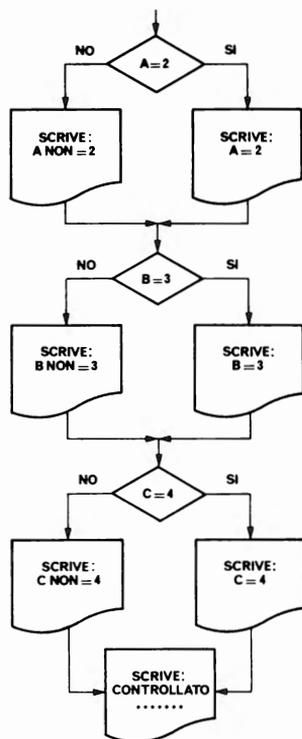
Esempi di frasi IF...THEN...

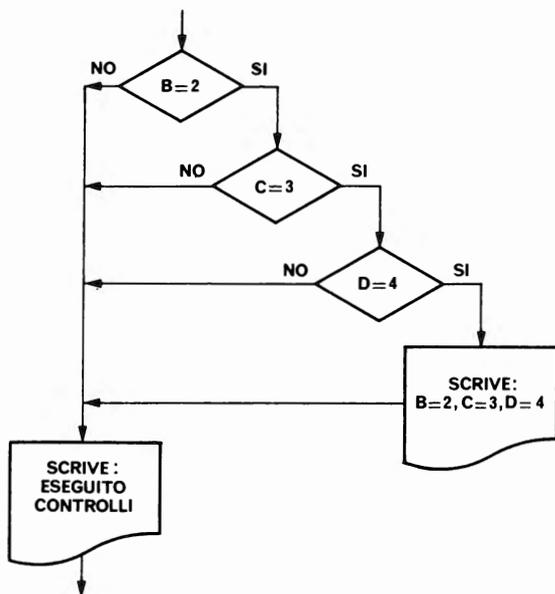
```
100 IF A = 3 THEN PRINT "A = 3"
120 PRINT "CONTROLLATO VALORE A"
```

Se A = 3 il programma quando arriva alla linea 100 esegue l'istruzione dopo il THEN e quindi scrive A = 3, dopo prosegue con la linea 120, quindi scri= ve CONTROLLATO VALORE A; se A e' diver= so da 3, va direttamente alla linea 120 e scrive solo CONTROLLATO VALORE A.
Quindi viene eseguita l'istruzione dopo THEN se la condizione risulta vera; e' eseguita solo la linea seguente se la condizione risulta falsa.
Le istruzioni 100 e 120 corrispondono al diagramma a blocchi qui a lato.

```
100 IF A = 3 THEN GOTO 150
120 PRINT "A DIVERSO DA 3"
130 PRINT "CONTROLLATO VALORE A"
140 GOTO......
150 PRINT "A RISULTA = 3"
160 GOTO 130
```

Se A = 3 il programma prosegue dalla linea 150, scrive:
A RISULTA = 3
e poi ritorna alla li= nea 130 e scrive:
CONTROLLATO VALORE A
e poi....; se A e' diverso da 3, il programma prosegue dalla linea 120 e scrive: A DIVERSO DA 3, e poi scrive: CONTROLLATO VALORE A, e poi....
Le istruzioni da 100 a 160 corrispondono al seguente diagramma a blocchi.

```
100 IF A = 2 THEN GOTO 130
110 PRINT "A NON = 2"
120 GOTO 140
130 PRINT "A = 2"
140 IF B = 3 THEN GOTO 170
150 PRINT "B NON = 3"
160 GOTO 175
170 PRINT "B = 3"
175 IF C = 4 THEN GOTO 200
180 PRINT "C NON = 4"
190 GOTO 210
200 PRINT "C = 4"
210 PRINT "CONTROLLATI VALORI A,B,C"
```

se A = 2 va a 130 e scrive: A = 2
se B = 3 va a 170 e scrive: B = 3
se C = 4 va a 200 e scrive: C = 4
poi scrive: CONTROLLATO........
se A diverso da 2, scrive: A NON = 2 e va a controllare B
se B diverso da 3, scrive B NON = 3 e va a controllare C
se C diverso da 4, scrive C NON = 4 e va a scrivere: CONTROLLATO.....

Le istruzioni da 100 a 210 corrispondono al seguente diagramma a blocchi.

```
100 IF B = 2 THEN IF C = 3 THEN IF D = 4 THEN
    PRINT "B = 2, C = 3, D = 4"
120 PRINT "ESEGUITO CONTROLLI"
```

Se B = 2, C = 3 e D = 4 il programma scrive:
B = 2, C = 3, D = 4
e dopo scrive: ESEGUITO CONTROLLI,
mentre se una delle 3 condizioni non e' verificata, scrive solo:
ESEGUITO CONTROLLI
Vale il diagramma a blocchi che segue.

Quest' ultima situazione viene chiamata degli "IF nidificati", si vedra' che puo' essere programmata anche in altro modo. Essa risulta piu' complicata delle situazioni logiche di diramazione viste precedentemente.

Sfruttando le relazioni con operatori logici, la 100 puo' essere scritta cosi':

```
100 IF B = 2 AND C = 3 AND D = 4 THEN PRINT "B = 2, C = 3, D
= 4"
```

e nel diagramma a blocchi si avrebbe un solo rombo con scritte all'interno tutte le condizioni che devono essere analizzate.

Per controllare i cicli nello ZX80 si usa una coppia di

istruzioni:

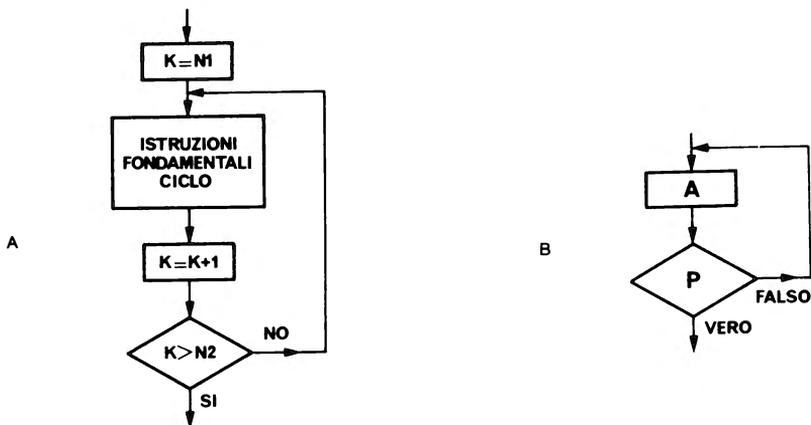
```
FOR variabile = espressione-1 TO espressione-2
...
...
NEXT variabile
```

dove "variabile" e' il nome della VARIABILE DI CONTROLLO del ciclo, e il suo nome puo' essere formato da una sola lettera; "espressione-1" puo' essere una costante, una variabile o una espressione aritmetica, il cui valore (sicuramente intero) rappresenta il valore iniziale della variabile di controllo del ciclo. Per "espressione-2" vale quanto detto per "espressione-1", ma essa rappresenta il valore finale che deve essere raggiunto dalla variabile di controllo del ciclo. Ad ogni ciclo la variabile di controllo viene incrementata di 1. Da cio' consegue che, per questo formato della istruzione FOR deve essere "espressione-1" minore di "espressione-2". Se esse sono uguali o se "espressione-1" supera a priori "espressione-2", il ciclo viene percorso una volta, infatti il controllo sulla variabile di controllo viene fatto al momento dell'esecuzione della istruzione NEXT. Questa istruzione puo' essere usata solo in modo differito. Le istruzioni comprese tra FOR e NEXT sono le "operazioni fondamentali del ciclo", cioe' quelle che devono essere ripetute. Esempio:

```
10 FOR K = N1 TO N2
20 PRINT K
30 NEXT K
```

Quando il programma arriva alla linea 10 viene posto K=N1, poi viene eseguita la linea 20 che stampa la variabile K. Al momento dell'esecuzione della linea 30 la variabile K viene incrementata di 1 e poi viene confrontata con N2. Se risulta K minore o uguale a N2 il programma torna alla linea 20, mentre se risulta K maggiore di N2 il programma va alla istruzione dopo la 30 ed il ciclo e' terminato. Per questa ragione all'uscita dal ciclo K = N2 + 1. E, sempre per questo modo di funzionamento, il ciclo viene sempre percorso almeno una volta.

```
                    FOR variabile = espressione-1 TO espressione-2
                             :                 :                   :
variabile di                 :                 :                   :
controllo     <-------                         :                   :
                                               :                   :
valore iniziale                                :                   :
variabile di controllo   <----------------------                   :
                                                                   :
valore finale                                                      :
variabile di controllo   <------------------------------------------
```

Il diagramma a blocchi della operazione iterativa per l'implementazione del Basic sullo ZX80 si presenta come nella illustrazione che segue, parte A. Nella stessa illustrazione, parte B, e' riportato il diagramma della situazione logica iterazione gia' visto. Dal confronto si constatano le differenze.

Il concetto di operazione ciclica e' fondamentale nella programmazione, si fa pero' notare che se non fosse disponibile la coppia di istruzioni FOR/NEXT si potrebbe ottenere lo stesso risultato gestendo a programma un contatore di ciclo ed usando la istruzione IF/THEN.

Nella implementazione del Basic valida sullo ZX81 e ZX80-Nuova ROM la coppia di istruzioni cicliche FOR/NEXT si scrive e lavora in modo diverso; vediamolo.

```
FOR variabile = espr-1 TO espr-2 STEP espr-3
...
...
NEXT variabile
```

dove "variabile" e' la VARIABILE DI CONTROLLO del ciclo e puo' avere il nome formato da una sola lettera; "espr-1", "espr-2" ed "espr-3" sono tre espressioni numeriche, intere o decimali. La prima rappresenta il valore iniziale della variabile di controllo, la seconda il valore finale della stessa e la terza l'incremento da dare alla variabile di controllo ad ogni ciclo. Inoltre al momento dell'esecuzione del FOR viene controllato se il ciclo puo' essere percorso almeno una volta, in caso contrario il ciclo non ha luogo. Puo' essere "espr-1" < "espr-2" e, in tale caso, "espr-3" deve essere un numero negativo. Al momento del NEXT viene

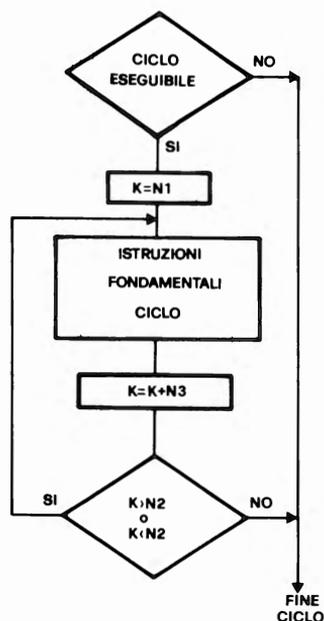
aggiunto alla variabile di controllo "espr-3" e poi essa viene controllata:

nel caso sia "espr-3" > = 0, per non maggiore del limite;
nel caso sia "espr-3" < 0, per non minore del limite;

prima di tornare ad eseguire le istruzioni fondamentali del ciclo. Se nella istruzione FOR si omette STEP, esso viene implicitamente assunto = 1. All'inizio dell'esecuzione della frase FOR il sistema cancella una eventuale variabile gia' in uso avente lo stesso nome, poi crea la variabile dandole il valore iniziale "espr-1". Come si vede la logica del FOR/NEXT e' abbastanza diversa nelle due implementazioni del Basic. Anche in questo caso la variabile di controllo all'uscita dal ciclo ha un valore che non e' stato usato nel ciclo, a seconda del segno di "espr-3" e' o maggiore o minore di "espr-2". In questo caso la FOR puo' essere usata anche in modo immediato; viene considerata istruzione fondamentale del ciclo la precedente.

La logica di questo FOR, che per comodita' di riferimenti scriviamo:

FOR K = N1 TO N2 STEP N3.............NEXT K

puo' essere rappresentata dal diagramma a blocchi che segue.

Si puo' fare uso di cicli nidificati, cioe' uno interno all'altro. Si veda l'esempio che segue. Si vuole evidenziare sullo schermo una tabella di 10 righe e di 10 colonne, nella quale la prima riga contiene tutti zeri, la seconda tutti uno, ecc.. Si possono scrivere le seguenti istruzioni:

```
10 FOR I = 0 TO 9
20 FOR K = 1 TO 10
30 PRINT I; " ";
40 NEXT K
50 PRINT
60 NEXT I
```

Il programma opera cosi':

. la linea 10 apre un ciclo FOR controllato dalla variabile I, ponendo I=0;

. la linea 20 apre un secondo ciclo FOR, interno al precedente, controllato dalla variabile K, ponendo K=1;

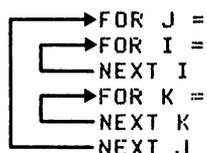
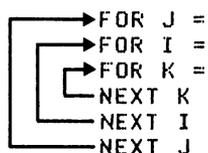
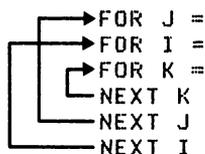
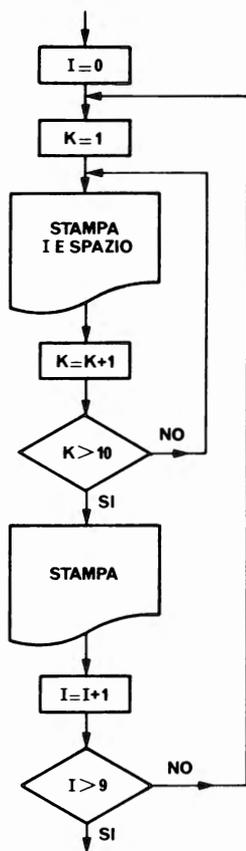
. la linea 30 e' l'istruzione fondamentale del ciclo controllato da K, essa stampa la variabile I seguita da uno spazio, senza andare a capo;

. la linea 40 incrementa K di 1 e controlla se ha superato il valore 10. Se K<=10 torna alla linea 30, se K>10 prosegue dalla linea 50;

. la linea 50 stampa una riga a vuoto per andare a capo;

. la linea 60 incrementa I e controlla se ha superato il valore 9. Se I>9 il programma termina, in caso contrario ritorna alla linea 20 e inizia nuovamente il ciclo FOR interno per stampare una nuova riga di numeri.

Il diagramma a blocchi della procedura e' quello della pagina seguente.

Quando si usano i cicli nidificati essi devono essere uno interno all'altro. Sono schemi corretti i seguenti:

Lo schema che segue invece non e' corretto:

L'istruzione di salto incondizionato ha la forma seguente:

GOTO num.linea

dove "num.linea" e' il numero della linea dalla quale si desidera far proseguire il programma. Puo' essere usata sia in modo differito che in modo immediato. Se per far partire il programma si usa GOTO n invece di RUN il programma inizia a lavorare senza avere prima azzerato le variabili. Se "num.linea" non esiste, il programma prosegue dalla prima linea che esiste con numero maggiore di "num.linea". Dopo il GOTO puo' comparire un numero, una variabile o una espressione. Nello ZX81 e ZX80-Nuova ROM se il numero non risulta intero, esso viene arrotondato all'intero piu' vicino.

L'istruzione per fermare un programma e':

STOP

essa va usata solo in modo differito. Causa la fermata del programma al numero di linea dello STOP. Per proseguire nell'esecuzione si deve usare il tasto CONT (CONTINUE). Al momento dello STOP viene segnalato errore di codice 9.

Del tasto BREAK si parla nel Capitolo 6.

Del gruppo di istruzioni di controllo fanno parte anche GOSUB e RETURN; di esse si parla nel paragrafo 5.15.

5.11. ISTRUZIONI PER L'INGRESSO E L'USCITA DEI DATI

L'istruzione per leggere dati dalla tastiera e':

INPUT nome-variabile

dove "nome-variabile" e' il nome della variabile dove si desidera memorizzare il dato che si scrive sulla tastiera. Il dato che viene scritto deve concordare con il tipo della variabile, cioe' non si deve rispondere con una stringa alfabetica alla richiesta di un numero.

Non si puo' usare il comando in modo immediato; se si vuole assegnare un dato ad una variabile in modo immediato si scrive: LET variabile = dato.

Quando il programma incontra questa istruzione si ferma in attesa di dati.

Sullo ZX80 se la variabile e' di tipo numerico, si vede il cursore sdoppiato con i due caratteri L ed S in campo inverso. Quando si scrive la prima cifra del numero scompare S; L scompare, insieme a tutto il numero, quando si preme NEW LINE per far accettare il dato. Se la variabile e' di tipo stringa il cursore con L in campo inverso appare tra due apici delimitatori e i dati immessi vengono scritti tra gli apici.Le stringhe possono essere lunghe a piacere compatibilmente con la capacita' della memoria. Se mentre si scrive una stringa scompare L e l'apice di chiusura, questo significa che si e' superato lo spazio disponibile. In questo caso si possono cancellare dei caratteri con SHIFT e RUBOUT fino a veder ricomparire il cursore e gli apici.

Sullo ZX81 e ZX80-Nuova ROM quando il calcolatore e' in attesa di INPUT il cursore evidenzia L per i numeri, "L" per le stringhe e resta nella parte bassa dello schermo.

Per quanto riguarda i dati numerici, si devono rispettare le regole dei due calcolatori.

L'istruzione per scrivere sul video e':

PRINT lista di variabili e/o costanti

dove i dati da stampare sono separati tra loro o da virgola o da punto e virgola. Il comando puo' essere usato anche in modo immediato. Sullo schermo sono disponibili 22 linee per scrivere le altre 2 servono per i comandi.
Le modalita' di esecuzione della PRINT differiscono nelle due versioni del Basic.

Per lo ZX80 i separatori tra i dati hanno il seguente effetto:

. la virgola fa posizionare alle colonne 9, 17 e 25 dello schermo, rendendo cosi' possibili 4 zone di stampa di 8 caratteri ciascuna, se un dato supera i 7 caratteri esso va a invadere la zona di stampa seguente e quindi la virgola fa saltare alla successiva, e se non c'e' piu' spazio sulla riga, manda a nuova riga. Due virgole vicine fanno saltare due zone di stampa.
. il punto e virgola fa stampare i dati senza caratteri separatori.

Se la lista dei dati da stampare termina con virgola o punto e virgola non si ha il salto a nuova riga, a meno che lo spazio sia terminato; l'effetto del tipo di separatore continua sulla nuova riga. Una riga tiene fino a 32 caratteri.
I numeri negativi vengono stampati preceduti dal segno meno. Nella lista dei dati da stampare possono comparire anche delle espressioni; esse vengono calcolate e viene stampato il risultato.

Per lo ZX81 e ZX80-Nuova ROM il comando PRINT si e' arricchito con le due funzioni AT e TAB; con la nuova ROM si puo' dire al calcolatore dove e cosa stampare sul video.

Dopo il comando PRINT si puo' scrivere una lista comprendente dati da stampare e funzioni di spostamento. Questi elementi devono essere separati dal punto e virgola. Due elementi da stampare possono anche essere separati dalla virgola, ma con la nuova ROM la virgola fa saltare di 16 posizioni (invece che di 8).
Questo non deve essere considerato un impoverimento dato che il comando TAB consente di andare dovunque sulla linea.
Vale ancora quanto detto per virgola e punto e virgola alla fine della lista dei dati da stampare.
Per il formato dei dati numerici valgono le seguenti regole:
. se il valore assoluto del numero e' compreso tra 10 elevato a -5 e 10 elevato a 13 (10**-5<X<10**13) esso viene

evidenziato nella normale notazione decimale con al massimo 8 cifre significative e senza zeri di riempimento dopo il punto decimale;

. se il valore assoluto del numero cade fuori del precedente intervallo, esso e' evidenziato in notazione scientifica, sempre con al massimo 8 cifre significative.

I numeri negativi sono preceduti dal segno meno.

Per lo ZX81 e lo ZX80-Nuova ROM e' disponibile un altro comando che riguarda il video:

SCROLL

esso sposta di una linea verso l'alto il contenuto dello schermo perdendo la linea superiore e posiziona la stampa all'inizio della linea disponibile in basso.

Esempi:

```
 5 REM PROVA COMANDO SCROLL
10 SCROLL
20 INPUT A$
30 PRINT A$
40 GOTO 10
```

```
 5 REM PROVA COMANDO TAB
10 FOR I=0 TO 20
20 PRINT TAB (8*I);I;
30 NEXT I
```

Il nuovo BASIC consente di usare una stampante collegata al calcolatore e fornisce tre istruzioni per comunicare con essa.

LPRINT

Questo comando consente di stampare dati con la stampante, corrisponde al comando PRINT per il video. Bisogna fare attenzione, se si vogliono usare AT e TAB, alle dimensioni orizzontali della linea di stampa che e' di 32 caratteri ed al fatto che nella funzione AT non viene considerata l'indicazione di linea, ma solo quella di colonna. La stampante ha un buffer per preparare la stampa delle dimensioni di 32 caratteri; la linea viene stampata:

. se il buffer e' pieno;

. se la lista dei dati dopo LPRINT non termina con virgola o punto e virgola;

. se la TAB manda a nuova linea;

. alla fine di un programma.

LLIST

Questo comando consente di mandare alla stampante liste di programmi, corrisponde al comando LIST per il video.

COPY

Questo comando trasferisce sulla stampante il contenuto del video.

Se volete fermare la stampante mentre lavora, potete usare il tasto BREAK.

5.12. ISTRUZIONI VARIE E DI SERVIZIO

CLEAR

Questa istruzione serve per cancellare tutte le variabili del programma liberando lo spazio che esse occupavano. Puo' essere usata anche in modo immediato.

CLS

Azzera lo schermo, puo' essere usata anche in modo immediato.

REM

Indica che quanto segue sulla linea e' un commento. Serve per inserire annotazioni in un programma, Non e' una istruzione operativa.

DIM nome-variabile (I1,I2,...In)

E' una istruzione di tipo dichiarativo e serve per creare variabili con indice riservando lo spazio necessario in memoria. Essa inizializza a zero le variabili numeriche e con spazi le variabili stringa. Si puo' usare anche in modo immediato. Tra parentesi devono essere indicate le dimensioni massime per ogni indice. Esistono notevoli differenze tra le due implementazioni del Basic riguardo alla DIM.
Nello ZX80 si possono dimensionare solo variabili numeriche, il cui nome e' formato da una sola lettera, ed esse possono avere un solo indice. Se si ridimensiona una variabile che esiste gia' il dimensionamento non ha effetto. Gli indici partono da 0; DIM A(6) crea una variabile A con 7

elementi.

Nello ZX81 e nello ZX80-Nuova ROM si possono dimensionare sia le variabili numeriche che le stringhe e sono consentite dimensioni multiple (paragrafo 5.7.). In questo caso viene accettato il ridimensionamento con cancellazione della variabile con indice generata con il dimensionamento precedente. Gli indici partono da 1; DIM A(6) crea una variabile A con 6 elementi.

RAND (RANDOMISE)

RAND 0 predispone il punto di partenza della sequenza dei numeri a caso ottenibili con la funzione RND ad un numero uguale al valore del contatore dei fotogrammi dello schermo. Se si scrive: RAND n, viene predisposto il punto di partenza della sequenza a n (n diverso da 0).

Puo' essere usata anche in modo immediato.

5.13. PEEK E POKE

La istruzione POKE e la funzione PEEK servono per intervenire direttamente sui byte di memoria tramite i loro indirizzi. Senza di esse non sarebbe possibile passare dal Basic al linguaggio macchina.

L'istruzione:

POKE a,b

serve per scrivere nel byte di indirizzo "a" l'espressione "b". "a" e "b" possono essere espressioni numeriche (in particolare costanti o variabili) e nello ZX80 devono essere intere, mentre nel nuovo Basic possono anche essere decimali e vengono arrotondate all'intero piu' vicino. Naturalmente essendo a l'indirizzo di un byte, questo deve essere compreso tra 0 e 32767 ed essere un indirizzo della RAM; mentre essendo b il contenuto di un byte, esso deve essere compreso tra 0 e 255.

La funzione:

PEEK (a)

serve per leggere il contenuto del byte di indirizzo a. Per "a" valgono le stesse considerazioni fatte sopra, ma puo' anche essere un indirizzo della ROM.

La memoria e' indirizzabile a byte, mentre le variabili intere usate dal Sistema Operativo sono contenute in due

byte consecutivi. In questo caso le cifre piu' significative della variabile si trovano nel byte di indirizzo dispari e le meno significative nel byte di indirizzo pari numericamente precedente. Per calcolare il valore del contatore dei fotogrammi dello schermo dello ZX80 che si trova nei byte 16414 e 16415 si deve procedere cosi':

```
10 LET C = PEEK (16414) + 256 * PEEK (16415)
20 PRINT C
```

5.14. LE FUNZIONI MATEMATICHE

Per lo ZX80 si hanno solo due funzioni di questo tipo; esse sono:

ABS (espressione)

che fornisce il valore assoluto di "espressione".

RND (espressione)

che fornisce un numero pseudo-random compreso tra 1 e "espressione" se questa e' positiva. Se "espressione" e' zero si ottiene il numero 1. Se "espressione" ha un valore negativo si ottiene un numero pseudo-random compreso tra -32767 e -1 oppure tra 1 e +32767. Ogni volta che viene usata la funzione RND il sistema usa un generatore di numeri a caso che produce una sequenza fissa di numeri, anche se tale sequenza e' molto lunga. Se prima di usare la RND si e' usata la RANDOMISE, viene preso come numero di partenza nella sequenza un numero uguale al valore del contatore dei fotogrammi dello schermo. Tale contatore inizia a contare quando viene acceso il sistema e viene incrementato di 1 ogni 50esimo di secondo. Il valore di questo contatore puo' essere alterato usando l'istruzione POKE. Se si e' premesso RANDOMISE si ottiene una diversa sequenza di numeri ogni volta che si fa girare il programma. Se invece si e' usato RANDOMISE n, con n diverso da zero, si ottiene di far partire la sequenza dei numeri da n e quindi, ogni volta che si fa girare il programma si ottiene la stessa sequenza.

Segue l'elenco delle funzioni matematiche disponibili sullo ZX81 e ZX80-Nuova ROM.

Tutte le funzioni, meno le due PI e RND, richiedono un argomento che non e' necessario porre tra parentesi se e' una costante o una variabile, ma va posto tra parentesi se e' una espressione.

Le funzioni di tipo matematico danno una precisione di circa 10 cifre e mantengono tali cifre in memoria anche se ne mostrano solo 8 sul video.

ELENCO FUNZIONI

Funz.	Argomento	Commento
ABS	numero	Valore assoluto.
ACS	numero	Arcocoseno in radianti. Errore A se argom. non tra -1 e +1.
ASN	numero	Arcoseno in radianti. Errore A se argom. non tra -1 e +1.
ATN	numero	Arcotangente in radianti.
COS	angolo in radianti	Coseno.
EXP	numero	Calcola "e" elevato al numero. e=2.718281828.
INT	numero	Parte intera del numero troncato senza arrotondamento.
LN	numero	Logaritmo naturale (base "e"). Errore A se l'argomento <=0.
PI	nessuno	Fornisce il numero PI=3.141592653 (p greco)
RND	nessuno	Fornisce il prossimo numero pseudo-random in una sequenza generata usando la formu= la: (75*(SEED+1)-1)/65536. SEED = al numero contenuto nel contatore dei foto= grammi dello schermo, ad altro se si e' usato il comando RAND. Il numero generato e'>=0 e <1.
SGN	numero	Fornisce: -1 se numero negativo 0 " " = 0 1 " " positivo.
SIN	angolo in radianti	Seno.
SQR	numero	Radice quadrata del numero. Errore B se numero negativo.
TAN	angolo in radianti	Tangente.

5.15. LE STRINGHE E LE FUNZIONI DI STRINGA

Nello ZX80 le stringhe possono essere solo variabili singole (senza indici); esse ricevono un contenuto o con una frase di assegnazione LET o con una frase di INPUT. Se nel corso del programma si cambiano i contenuti di una stringa, cioe' essa compare a sinistra di un = o dopo INPUT, anche se le sue dimensioni non variano, ne viene creata una nuova e lo spazio che si era usato viene riutilizzato spostando in su tutte le altre variabili e ricreando la stringa in coda.

Le funzioni di stringa disponibili sono 4 e precisamente:

CHR$ (espr)

dove "espr" e' una espressione numerica intera e deve essere compresa tra 0 e 255. La funzione fornisce il carattere ASCII corrispondente al valore di "espr". Vedi Appendice A.

TL$ (stringa)

dove "stringa" e' una qualunque stringa. La funzione fornisce una nuova stringa ottenuta dalla precedente privandola del primo carattere. Se scriviamo:
10 PRINT TL$ ("ABCDE") otteniamo sul video BCDE.

CODE (stringa)

fornisce il codice numerico corrispondente al primo carattere della stringa. "stringa" puo' essere una costante o una variabile. Se scriviamo:
10 PRINT CODE ("OGGI PIOVE") otteniamo 52 che e' il codice di O.

STR$ (espr)

fornisce una stringa di caratteri corrispondente al valore di "espr". Esempio:

```
10 LET A$ = STR$ (4567)   pone A$ = "4567"
20 LET A$ = STR$ (-23)    pone A$ = "-23"
```

Vediamo ora il trattamento delle stringhe nello ZX81 e ZX80-Nuova ROM.

Le variabili stringa ricevono un contenuto o con una frase LET di assegnazione o con una INPUT. Nel primo caso il dato deve essere contenuto tra doppi apici. Ovviamente l'unico carattere che non puo' far parte della stringa e' il doppio apice (", chiamato QUOTE), ma si puo' ottenerlo, se desiderato, usando il carattere chiamato "QUOTE IMAGE" corrispondente ai tasti SHIFT e Q (""), il quale in fase di

stampa appare come un doppio apice. Per quanto riguarda il comportamento in memoria vale quanto detto precedentemente per lo ZX80 riguardo alle stringhe non dimensionate. Le stringhe con indice invece sono fissate dalla frase DIM e non vengono spostate quando ricevono nuovi contenuti; esse sono di lunghezza predeterminata.

Si puo' usare l'operatore + per concatenare tra loro piu' stringhe, cioe'

```
10 LET A$ = "GIORNATA "
20 LET B$ = "DI FESTA"
30 LET C$ = A$ + B$
40 PRINT C$
```

appare GIORNATA DI FESTA perche' C$ contiene le due stringhe A$ e B$ concatenate.

Nel seguito vengono elencate le funzioni che hanno attinenza con il trattamento delle stringhe. Per ognuna viene indicato il tipo dell'argomento; esso deve essere scritto tra parentesi solo se e' una espressione. Se e' una costante o una variabile puo' essere scritto senza parentesi.

Le funzioni disponibili per le stringhe sono:

.CHR$ (argomento numero)
Fornisce il carattere corrispondente al codice numerico su cui opera. Il codice deve essere compreso tra 0 e 255, altrimenti si ha errore.
Esempio:

```
10 FOR K=1 TO 26
20 PRINT CHR$(K+37);
30 NEXT K
```

stampa le 26 lettere dell'alfabeto.

.CODE (argomento stringa)
Fornisce il codice numerico del primo carattere della stringa. Se la stringa e' la stringa nulla ottenuta scrivendo due volte il doppio apice (da non confondere con il carattere SHIFT e Q) si ot=tiene 0. Esempio:

```
100 PRINT CODE("OGGI")
```

stampa 52, codice della lettera O.

.LEN (argomento stringa)
Fornisce la lunghezza della stringa. Se appli=cata alla stringa nulla da' 0. Esempio:

```
20 LET X = LEN (A$)
```

se A$="PIOVE", pone in X il valore 5.

.STR$ (argomento numero)
Trasforma un numero o una espressione nella stringa corrispondente. Esempio:

```
10 LET C=-345
20 PRINT STR$(C),STR$(34+8.9)
```

stampa

```
-345            42.9
```

.VAL (argomento stringa)
Fornisce un numero corrispondente alla stringa che deve essere numerica, altrimenti si ha er= rore. Esempio:

```
20 LET A$="-345.8"
30 LET X=VAL(A$)
40 LET Z = X + 18
```

consente di operare un calcolo sul contenuto di A$.

Non e' piu' disponibile la funzione di stringa TL$, ma essa non e' piu' necessaria potendo trattare una qualunque stringa come una variabile stringa con indice e quindi potendo accedere ad ogni carattere mediante un indice.

Esempio:

```
LET A$="ABCDEFGHILMNOPQ"
LET B$=A$(2)    pone B$="B"
LET B$=A$(9)    pone B$="I"
```

Si definisce SUBSTRINGA una qualunque porzione di STRINGA formata da caratteri consecutivi. Se consideriamo la stringa A$="FELICEMENTE", la stringa B$="MENTE" e' una substringa di A$, mentre la stringa C$="LIMENTE" non lo e' perche' non e' formata tutta da caratteri consecutivi di A$.
Nel nuovo BASIC c'e' la possibilita' di riferirsi a substringhe di una qualunque stringa.
Per ottenere la stringa B$ di cui sopra possiamo scrivere:

```
100 LET B$ = A$(7 TO 11)
```

cioe' prendiamo i caratteri di A$ dal settimo all'undicesimo.
Questo tipo di operazione prende il nome di "slicing". Si deve far seguire alla stringa dalla quale si vuole estrarre una parte una coppia di parentesi e porre entro le parentesi il numero d'ordine del carattere da cui iniziare l'estrazione, la parola chiave TO ed il numero d'ordine del carattere con il quale terminare l'estrazione. Uno dei due numeri o tutti e due possono mancare, come risulta dagli esempi seguenti, che non sono scritti nella forma di frasi BASIC, ma servono solo per spiegare la logica dell'operazione:

```
"PIPPO"( TO 5) = "PIPPO"(1 TO 5) = "PIPPO"
"PIPPO"(2 TO ) = "PIPPO"(2 TO 5) = "IPPO"
```

```
"PIPPO"( TO )  = "PIPPO"(1 TO 5) = "PIPPO"
"PIOVE"(2 TO 2) = "PIPPO"(2) = "I"
"PIOVE"(3 TO 7) da' errore, la stringa e' di 5 caratteri
"PIPPO"(5 TO 4) = "", cioe' la stringa nulla.
```

I due numeri devono essere positivi, altrimenti si ha errore.

Il programma che segue toglie dalla stringa A$ tutti gli spazi di riempimento a destra, ottenendo una stringa B$, e poi stampa le due stringhe tra doppi apici.

```
10 INPUT A$
20 FOR N=LEN A$ TO 1 STEP -1
30 IF A$(N)<>" " THEN GOTO 50
40 NEXT N
50 LET B$ = A$(TO N)
60 PRINT """";A$;"""","""";B$;""""""
70 GOTO 10
```

Alla linea 30 l'operazione di "slicing" consente di trattare i caratteri della stringa A$ come se essa fosse una stringa dimensionata con una DIM come variabile con indice. Alla linea 60 si fa uso del carattere "quote image" per ottenere la stampa delle due stringhe A$ e B$ tra doppi apici. Se la stringa A$ fosse tutta di spazi, alla linea 50 si arriverebbe con N=0 e quindi B$ risulterebbe la stringa nulla.

Se si opera su variabili stringa, ed ovviamente non su costanti, si possono anche modificare alcuni caratteri nella stringa, cioe' operare una sostituzione invece di una estrazione. Esempio:

```
10 LET A$="SEI FELICE"
20 LET A$(5 TO 10)="******"
30 PRINT A$
```

si ottiene: SEI ******

Se alla linea 20 la substringa sostitutiva e' piu' lunga della parte da sostituire essa viene troncata.

L'operazione di "slicing" ha priorita' 12.

L'operazione di "slicing" non e' standard; essa e' molto versatile e consente di supplire alla mancanza in questo Basic di funzioni di stringa come: LEFT$, RIGHT$ e MID$.

5.16. FUNZIONI VARIE

In tutte le due versioni del Basic e' presente la funzione USR che permette di andare ad eseguire un programma in linguaggio macchina.
La funzione si scrive cosi':

USR (numero)

dove numero deve essere un numero intero per lo ZX80 e nell'altro caso viene arrotondato all'intero piu' vicino. Tale numero rappresenta l'indirizzo del byte a partire dal quale e' stato memorizzato il programma in linguaggio macchina. La funzione fornisce un risultato che e' precisamente il contenuto dei registri HL per il vecchio Basic e BC per il nuovo Basic, se il contenuto di tali registri e' stato modificato a causa dell'esecuzione del programma in codice macchina. Se tale contenuto non e' stato modificato ritorna il numero usato nella chiamata.

Si descrivono tutte le altre funzioni valide per lo ZX81 e lo ZX80-Nuova ROM.

Funz.	Argomento	Commento
AT	numeri	L'argomento e' dato da due numeri sepa= rati da virgola: ATx,y, dove x e y rap= presentano le coordinate del punto del video dove si vuole evidenziare il pros= simo carattere. Il primo numero, x, si riferisce alla linea e puo' variare da 0 a 21. Il secondo numero, y, si riferi= sce alla colonna e puo' variare da 0 a 31. Questa funzione puo' essere usata nei comandi PRINT e LPRINT. La linea 0 e' la piu' alta e colonna 0 la piu' a sini= stra. Il video appare come se si disegna il primo quadrante degli assi cartesiani ponendo l'origine nell'angolo in alto a sinistra, l'asse x dall' alto verso il basso e l'asse y orizzontale orientato da sinistra a destra. Rispetto agli assi usati dalla PLOT si ha una traslazione verso l'alto ed una rotazione di 90 gra= di in senso orario. Con LPRINT non viene considerata l'indicazione di linea. Dopo AT l'elemento seguente deve essere pre= ceduto dal punto e virgola.

INKEY$	nessuno	Legge un carattere dalla tastiera, esso corrisponde al tasto premuto quando il cursore e' nel modo L. Se non si preme alcun tasto si ha la stringa nulla (si veda paragrafo 9.24.).
NOT	relazio= ne logica	Se NOT relazione logica e' vero la varia= bile logica e' = 1, altrimenti e' = 0.
TAB	numero	Sposta la posizione di stampa alla co= lonna indicata dall'argomento. Se il nu= mero e' maggiore di 31, la funzione lavo= ra sul resto del numero diviso 32. La linea non viene variata a meno che la colonna richiesta comporti uno spostamen= to all' indietro. La posizione 0 e' la piu' a sinistra sulla linea. TAB puo' es= sere usata con PRINT e LPRINT.

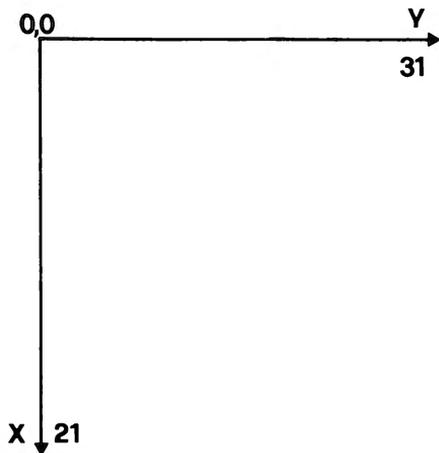



Fig. 5.1. Assi usati da AT

Fig. 5.2. AT e TAB con LPRINT

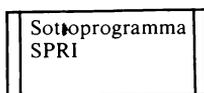
5.17. I SOTTOPROGRAMMI

Con il termine SOTTOPROGRAMMA si intende un pezzo di programma concluso in se stesso e che svolga una determinata funzione. Tale sottoprogramma deve poter essere usato da parte di altri programmi richiamandolo o inserendolo nel contesto. In dipendenza dal linguaggio di programmazione usato si puo' avere la possibilita' di memorizzare su un supporto i sottoprogrammi e di richiamarli nel programma principale, lasciando al sistema la cura di inserirli materialmente. In questo caso si parla di sottoprogrammi esterni al programma principale. Con altri linguaggi di programmazione, come il Basic, si ha solo la possibilita' di inserire materialmente nel listato del programma principale i sottoprogrammi; si parla in questo caso di sottoprogrammi interni. Con altri linguaggi di programmazione si hanno ambedue le possibilita'.

La tecnica dell'uso dei sottoprogrammi e' molto utile perche' consente di programmare con minore fatica e con maggiore chiarezza. Una volta che un sottoprogramma e' stato provato, esso puo' essere inserito in qualunque programma per ottenere gli stessi risultati. Inoltre, se in un programma si devono rifare in punti diversi le stesse sequenze di operazioni conviene scriverle una sola volta come sottoprogramma e richiamarle dai diversi punti.

Da quanto detto risulta che nel linguaggio devono essere disponibili le seguenti istruzioni:

. una istruzione per saltare all'inizio del sottoprogramma interno, memorizzando il numero di linea successivo a quello della linea che contiene l'istruzione di salto;

. una istruzione con la quale chiudere il sottoprogramma interno e ritornare alla sequenza principale al numero di linea precedentemente memorizzato.

Nella stesura dei diagrammi a blocchi si usa questo simbolo grafico per indicare la chiamata ad un sottoprogramma. Si scrive internamente il nome del sotto= programma chiamato e se ne trac= cia a parte il diagramma.

Le due istruzioni di cui sopra, sono:

GOSUB num.-linea essa serve per saltare al sottoprogram= ma che inizia in num.-linea e per memo=

	rizzare il numero di linea seguente la istruzione GOSUB.
RETURN	per chiudere il sottoprogramma logica=mente e fare ritornare al programma nel punto giusto.

Nel programma esempio che segue si esemplifica cosa e' un sottoprogramma:

```
 10 REM PROVA SOTTOPROGRAMMA
 20 REM PRIMA CHIAMATA
 30 GOSUB 500
 40 PRINT "SONO TORNATO LA PRIMA VOLTA"
 50 REM SECONDA CHIAMATA
 60 GOSUB 500
 70 PRINT "SONO TORNATO ANCORA"
 80 STOP
 ....
 ....
500 REM SOTTOPROGRAMMA PROVA
510 PRINT "SONO UN SOTTOPROGRAMMA"
520 RETURN
```

Dopo aver fatto girare il programma si vedra' sul video:

```
SONO UN SOTTOPROGRAMMA
SONO TORNATO LA PRIMA VOLTA
SONO UN SOTTOPROGRAMMA
SONO TORNATO ANCORA
```

La sequenza di esecuzione delle linee di programma e' stata la seguente:

10 - 20 - 30
500 - 510 - 520
40 - 50 - 60
500 - 510 - 520
70 - 80

La istruzione GOSUB puo' essere usata sia in modo immediato che differito; la RETURN non ha senso se usata in modo immediato.

Si consiglia di attribuire numeri bassi di linea ai sottoprogrammi, dato che il sistema, quando incontra GOSUB inizia a ricercare il numero di linea partendo dalla prima linea di programma. Si puo' iniziare il programma con:

```
01 GOTO 1000
```

far seguire i sottoprogrammi e da 1000 in poi mettere il programma principale.

5.18. IL CONTROLLO DEL TEMPO

Nello ZX81 e nello ZX80-Nuova ROM e' possibile programmare delle attese calcolate servendosi del comando PAUSE. Si scrive:

```
PAUSE n
```

e il programma si ferma per un intervallo di tempo pari al tempo necessario per far apparire n fotogrammi sul video. La velocita' dei fotogrammi e' di 50 al secondo; con n=32767 si ottiene una pausa di circa 11 minuti. Se n e' maggiore di 32767 la pausa corrisponde allo STOP. Si puo' interrompere la pausa premendo un qualunque tasto.

Al comando PAUSE si deve far seguire una POKE particolare; si deve quindi scrivere:

```
PAUSE n
POKE 16437,255
```

questa POKE serve a riposizionare il byte alto del contatore dei fotogrammi. Non e' necessario usare questa POKE se si lavora con lo ZX81 in modo SLOW.

Con il programma che segue si ottiene un orologio funzionante sul video.

```
 5 REM DISEGNAMO L'OROLOGIO
10 FOR N=1 TO 12
20 PRINT AT 10-10*COS(N/6*PI),10+10*SIN(N/6*PI);N
30 NEXT N
35 REM FACCIAMO PARTIRE L'OROLOGIO
40 FOR T=0 TO 10000
45 REM T E' IL TEMPO IN SECONDI
50 LET A=T/30*PI
60 LET SX=21+18*SIN A
70 LET SY=22+18*COS A
75 PLOT SX,SY
77 PAUSE 42
79 POKE 16437,255
81 UNPLOT SX,SY
90 NEXT T
```

Le attese non calcolate si ottengono usando il comando

STOP e poi CONT per proseguire.

```
  5 REM DISEGNO OROLOGIO
 10 FOR N=1 TO 12
 20 PRINT AT 10-10*COS (N/6*PI)
,10+10*SIN (N/6*PI);N
 30 NEXT N
 35 REM PARTENZA OROLOGIO
 40 FOR T=0 TO 10000
 45 REM T=TEMPO IN SECONDI
 50 LET A=T/30*PI
 60 LET SX=21+16*SIN A
 70 LET SY=22+16*COS A
 75 PLOT SX,SY
 77 PAUSE 42
 79 POKE 16437,255
 81 UNPLOT SX,SY
 90 NEXT T
```

```
          12
     11         1

  10               2

 9                  3
                    ▪
  8                4

     7          5
          6
```

Fig. 5.3. Lista su stampante   Fig. 5.4. Orologio con COPY

Si puo' usare il comando INKEY$ per ottenere delle attese controllandone la durata esternamente al programma. Infatti il comando INKEY$ legge dalla tastiera un carattere, se non si preme alcun tasto legge la stringa nulla. Premendo un qualunque tasto e controllandolo a programma si generano delle attese. Il programma che segue prosegue solo se si preme un tasto qualunque:

```
10 IF INKEY$ = "" THEN GOTO 10
20 ........
```

infatti se non si preme alcun tasto e quindi viene letta la stringa nulla la linea 10 ritorna su se stessa (si veda paragrafo 9.24.).

Il programma che segue si ferma fino a quando si preme un tasto, se esso e' A prosegue dalla linea 500, se altro prosegue dalla linea 100:

```
10 IF INKEY$ = "" THEN GOTO 10
20 IF INKEY$ = "A" THEN GOTO 500
30 GOTO 100
```

5.19. LA GRAFICA

Questo paragrafo si riferisce allo ZX81 ed allo ZX80-Nuova ROM.

Lo schermo fornisce di norma 22*32 = 704 posizioni di

stampa (sono state escluse le ultime due linee). Con i comandi della grafica ognuno di questi 704 punti puo' essere ulteriormente suddiviso in 4 puntini (PIXEL).

Ogni "puntino" ha due coordinate, x e y, che lo individuano. Queste coordinate si scrivono abitualmente entro parentesi, cosi': (5,7); in questo caso si intende riferire un puntino che dista 5 dall'estrema sinistra dello schermo e 7 dal basso. Le coordinate dei puntini negli angoli dello schermo, girando in senso antiorario e partendo dall'angolo in basso a sinistra, sono rispettivamente:

(0,0),(63,0),(63,43),(0,43).

Fig. 5.5 Assi usati da PLOT e UNPLOT

I comandi disponibili sono:

PLOT x,y	scrive un puntino nella posizione x,y
UNPLOT x,y	cancella il puntino nella posizione x,y.

Si deve fare attenzione al fatto che le coordinate dei puntini nei comandi PLOT e UNPLOT sono trattate in modo inverso rispetto alla funzione AT.
Nella funzione AT le linee sono numerate da 0 a 21 muovendosi dall'alto verso il basso, e le colonne sono numerate da 0 a 31 muovendosi da sinistra verso destra. Inoltre il primo numero si riferisce alle linee e il secondo alle colonne.

Nei comandi PLOT e UNPLOT le coordinate dei puntini vanno da 0 a 43 muovendosi dal basso verso l'alto e da 0 a 63 muovendosi da sinistra verso destra. Inoltre la prima coordinata si riferisce alle colonne e la seconda alle

linee.

Esempi:

```
10 REM GRAFICO FUNZIONE SENO
15 REM TRA 0 E 2PI
20 FOR N = 0 TO 63
30 PLOT N,22+20*SIN(N/32*PI)
40 NEXT N
```

```
10 REM DISEGNA PUNTI A CASO OGNI
20 REM VOLTA CHE SI PREME NEWLINE
30 PLOT INT(RND*64),INT(RND*44)
40 INPUT A$
50 GOTO 30
```

```
10 REM GRAFICO DI SQR TRA 0 E 4
20 FOR N = 0 TO 63
30 PLOT N,20*SQR(N/16)
40 NEXT N
```

Segue un sottoprogramma che traccia una linea tra due puntini; le coordinate dei due puntini devono essere lette dalla tastiera nel programma principale.

Le coordinate siano (A,B) e (C,D).

```
1000 LET U=C-A
1005 REM U=NUMERO PASSI ORIZZONTALI
1010 LET V=D-B
1015 REM V=NUMERO PASSI VERTICALI
1020 LET D1X=SGN U
1030 LET D1Y=SGN V
1035 REM D1X E D1Y SONO UNO SPOSTAMENTO LUNGO
1036 REM LA DIAGONALE
1040 LET D2X=SGN U
1050 LET D2Y=0
1055 REM D2X E D2Y SONO UNO SPOSTAMENTO VERSO
1056 REM DESTRA O VERSO SINISTRA
1060 LET M=ABS U
1070 LET N=ABS V
1080 IF M>N THEN GOTO 1130
1090 LET D2X=0
1100 LET D2Y=SGN V
1105 REM D2X E D2Y SONO UNO SPOSTAMENTO VERSO
1106 REM L'ALTO O VERSO IL BASSO
1110 LET M=ABS V
1120 LET N=ABS U
1130 REM M E' IL MAGGIORE TRA ABSU E ABSV
1140 LET S=INT(M/2)
1145 REM VOGLIAMO MUOVERCI DA (A,B) A (C,D) IN M PASSI
1146 REM USANDO: N VOLTE L'INCREMENTO D2 PER SPOSTAMENTI
```

```
1147 REM ORIZZONTALI E VERTICALI E M-N VOLTE L'INCREMENTO
1148 REM D1 PER SPOSTAMENTI DIAGONALI, DISTRIBUITI IL PIU'
1149 REM UNIFORMEMENTE POSSIBILE
1150 FOR I = 1 TO M
1160 PLOT A,B
1170 LET S=S+N
1180 IF S<M THEN GOTO 1230
1190 LET S=S-M
1200 LET A=A+D1X
1210 LET B=B+D1Y
1215 REM SPOSTAMENTO DIAGONALE
1220 GOTO 1250
1230 LET A=A+D2X
1240 LET B=B+D2Y
1245 REM SPOSTAMENTO ORIZZONTALE O VERTICALE
1250 NEXT I
1260 RETURN
```

5.20. FAST E SLOW

Lo ZX81 ha due comandi che mancano anche sullo ZX80-Nuova ROM; essi sono:

FAST e SLOW

e sono disponibili sulla tastiera.

Questo calcolatore ha la possibilita' di funzionare con due diverse velocita'. Al momento dell'accensione esso si trova nel modo SLOW e, in tale modo, la velocita' di calcolo e' minore, ma lo schermo resta sempre attivo e non spariscono i suoi contenuti, come succede nello ZX80 e nello ZX80-Nuova ROM. Infatti questi due calcolatori possono funzionare sempre e solo in modo FAST.
Il modo SLOW e' l'ideale per fare della grafica, mentre se si devono fare lunghi calcoli e' megli passare nel modo FAST.
Il passaggio da un modo all'altro si ottiene molto semplicemente, sia in modo immediato che differito scrivendo o FAST o SLOW.

Potete provare ad introdurre in un qualunque programma il comando FAST, farlo girare, e poi modificare il comando in SLOW e farlo girare un altra volta e vedrete la differenza. Il comando di cambio velocita' puo' anche essere dato in modo immediato prima di fare girare il programma.

In fase caricamento programmi da tastiera si consiglia di lavorare in modo FAST. Provate con un programma che superi le 22 linee e vi renderete conto del perche' di questo

consiglio.

CAPITOLO 6

C O M E   O P E R A R E

6.1. LE SEGNALAZIONI SUL VIDEO

Sullo schermo si hanno due indicatori. Uno e' il CURSORE DELLO SCHERMO e l'altro il PUNTATORE DI LINEA. Essi sono due quadratini neri, nei quali appaiono i caratteri in bianco, cioe' in campo inverso.

Sullo schermo si possono avere 24 linee di 32 caratteri ciascuna, ma le due linee in basso sono riservate ai comandi.
Il calcolatore puo' funzionare in due modi:

. sotto controllo del sistema;
. sotto controllo del programma.

Dopo l'accensione e la sintonizzazione sul cursore nell'angolo in basso a sinistra compare K a indicare che il calcolatore e' sotto controllo del sistema e puo' accettare solo comandi (parole chiave usate con la giusta sintassi). Dopo aver caricato un programma e fatta partire l'esecuzione dello stesso il calcolatore lavora sotto controllo del programma e restituisce il controllo al sistema o quando il programma e' terminato o quando si incontra uno STOP o quando si ha una segnalazione di errore. Se si lavora in modo immediato, dopo l'esecuzione di ogni istruzione il calcolatore torna sotto controllo del sistema.

Vediamo ora i possibili contenuti del cursore quando il calcolatore si trova sotto controllo del sistema. La lettera che compare nel cursore influenza l'interpretazione che il sistema da' alla pressione dei tasti consentendo di usare lo stesso tasto per scopi molteplici.

I contenuti del cursore possono essere:

. K se in attesa di comando;
. L se in attesa di carattere;
. F (solo per ZX81 e ZX80-Nuova ROM) in attesa di funzione;
. G (solo per ZX81 e ZX80-Nuova ROM) in attesa di carattere grafico o di carattere in campo in=verso.

Gli stati K ed L non possono essere determinati dall'utente, mentre lo stato F si produce con la pressione contemporanea di SHIFT e FUNCTION e resta attivo solo per la pressione del tasto successivo. La lettera G compare se si premono contemporaneamente SHIFT e GRAPHICS , resta attiva fino a quando si premono di nuovo contemporaneamente questi due tasti e consente di selezionare:

. un carattere in campo inverso premendo il relativo tasto;

. un carattere grafico premendo il relativo tasto contemporaneamente allo SHIFT.

Non e' corretto passare allo stato G se il cursore si trovava in attesa di comandi, stato K. Il sistema accetta lo stato G, ma poi non accetta la linea di programma e segnala errore. E' corretto passare allo stato G se si era nello stato L.

Durante il caricamento di linee di programma o di comandi in modo immediato, se si commettono degli errori il cursore si sdoppia in due cursori, con S prima dell'errore ed L dopo. Una linea con errori non viene accettata alla pressione del tasto NEW LINE. Per correggere gli errori si puo' spostare il cursore verso destra o sinistra servendosi dei due tasti appositi (SHIFT e 8 - SHIFT e 5) e si possono cancellare gli errori usando SHIFT e RUBOUT. Le linee di programma si formano nella parte bassa dello schermo e salgono quando vengono accettate.

Durante l'esecuzione di un programma il cursore dello schermo segnala l'attesa di INPUT in questo modo:

. nello ZX80 salendo alla posizione libera dello schermo e sdoppiandosi in due cursori con L a sinistra ed S a destra se attende dati numerici o L tra apici se in attesa di stringa;

. nello ZX81 e ZX80-Nuova ROM restando nell'angolo a sinistra in basso e mostrando L se in attesa di numeri e "L" se in attesa di stringa.

Il puntatore di linea contiene sempre il carattere maggiore (>); esso puo' essere spostato in giu' e in su usando i relativi tasti (SHIFT e 6 - SHIFT e 7). Il puntatore di linea compare sul video quando si fa accettare la prima linea di programma; esso punta sempre l'ultima linea di programma introdotta.

Sullo ZX80 il tasto HOME (SHIFT e 9) agisce sul puntatore di linea facendolo salire alla linea 0. Dal momento che la linea zero non esiste sullo schermo, usando HOME il puntatore di linea svanisce; per farlo ricomparire basta usare il tasto freccia-giu' (SHIFT e 6).

Quando si usa il comando LIST ed appare il programma sullo schermo il puntatore di linea non e' presente; se si usa il tasto freccia-giu' esso ricompare.

6.2. IMMISSIONE DI UN PROGRAMMA

Prima di scrivere un nuovo programma premere il tasto NEW e poi NEW LINE per azzerare la memoria. Il cursore dello schermo si pone al valore K.

Le linee di programma si scrivono usando i tasti appropriati e si vedono formare nella parte bassa dello schermo; il cursore segue la scrittura della linea, cambiando di stato e segnalando eventuali errori. Quando la linea e' completa il tasto NEW LINE la fa accettare solo se non ci sono errori formali; se ci sono errori la linea rimane nella parte bassa dello schermo. In questo caso si muove opportunamente il cursore e si cancellano gli errori usando il tasto RUBOUT (SHIFT e 0). Si deve tener presente che RUBOUT cancella quello che e' scritto a sinistra del cursore; se si cancella un carattere normale, viene cancellato un solo carattere, se si cancella una parola chiave, essa viene completamente cancellata.

Se si vuole inserire un carattere, basta usare il tasto appropriato ed il carattere viene inserito a sinistra del cursore spostando tutta la linea verso destra. Lo spostamento e' di una posizione per inserimento di caratteri normali, di tutte le posizioni necessarie per inserimento di parole chiave.

Quando la linea e' tutta corretta essa viene accettata premendo NEW LINE e passa nella parte alta dello schermo nella posizione che le compete in base al numero di linea, con il puntatore di linea posizionato subito dopo il numero di linea. Se nella lista del programma esisteva gia' una linea con lo stesso numero della nuova, la vecchia linea viene cancellata ed al suo posto va la nuova.

Una linea di programma gia' accettata puo' necessitare di correzioni per errori logici o di simboli creati dal programmatore e non contrastanti con la sintassi del linguaggio. In tale caso si puo' procedere cosi':

. si sposta il puntatore di linea alla linea voluta usando i due tasti SHIFT e 6 o SHIFT e 7;
. si usa il tasto EDIT, questo fa comparire la linea nella parte bassa dello schermo;
. spostando il cursore dello schermo per mezzo dei tasti

SHIFT e 5 o SHIFT e 8, usando SHIFT e 0 (RUBOUT) ed i tasti appropriati, si modifica la linea;

. premendo NEW LINE la linea modificata va a sostituire la vecchia nella lista del programma.

Questa procedura di EDIT puo' essere utilmente impiegata qualora in un programma si abbiano linee uguali a meno del numero di linea, o, comunque, abbastanza simili tra loro.

Quando il programma supera le 22 linee sullo schermo, ad ogni nuova linea aggiunta si ha la perdita apparente delle prime linee. Queste linee scompaiono solo dallo schermo, ma restano in memoria. Per far comparire la lista dall'inizio basta usare il tasto LIST. Questo comando e' descritto nel paragrafo 5.5.; si ricorda che con LIST si ha la lista dall'inizio per le linee che entrano nello schermo, mentre con LIST n, si ha la lista dalla linea n in avanti.

Se si desidera cancellare una linea di programma, si deve scrivere il numero della linea e subito dopo premere NEW LINE. Se si scrive il numero della linea seguito da uno o piu' spazi e poi NEW LINE, la vecchia linea viene sostituita dalla nuova, contenente solo il numero di linea e questa non disturba durante durante l'esecuzione del programma.

6.3. ESECUZIONE DI UN PROGRAMMA

Per mandare in esecuzione un programma si usa il tasto RUN. L'effetto di RUN e' quello di azzerare tutte le variabili del programma e di farne partire l'esecuzione dal numero di linea minore.

Qualora si desideri far partire un programma dalla linea N, azzerando prima le variabili, si scrive RUN N.

Se invece si vuole mandare in esecuzione un programma senza azzerare le variabili si deve scrivere: GOTO N, dove N e' il numero o della prima linea del programma o della linea dalla quale si vuole partire.

Nello ZX80, nello ZX80-Nuova ROM e nello ZX81 in stato FAST mentre il programma lavora lo schermo si oscura e scompaiono le scritte. Se invece si usa lo ZX81 in stato SLOW si ha la persistenza delle scritte sul video, ma il calcolatore lavora piu' lentamente.

Il programma non puo' essere interrotto quando e' in attesa di INPUT. Nel paragrafo 6.1. e' descritto il comportamento del cursore quando il calcolatore e' in attesa di dati. Se si ha un errore nei dati, appare la segnalazione di errore. In questo caso si puo' ripartire dal punto voluto

con GOTO N.
Sullo schermo restano tutti i dati prodotti dalle istruzioni PRINT.

Alla conclusione del programma o ad una sua possibile interruzione si ha la segnalazione di errore, che potrebbe anche non essere un errore, nell'angolo in basso a sinistra dello schermo e viene indicato anche il numero della linea di programma eseguita per ultima.
La segnalazione degli errori ha il seguente formato: n/m, dove:

. n = numero dell'errore;
. m = numero di linea del programma.

Quando il programma ha, per una qualunque ragione, restituito il controllo al sistema, se nello ZX80 si preme un qualunque tasto riappare la lista del programma. Se nel programma sono stati inseriti degli STOP, per continuare basta usare il tasto CONT. Solo che appena si tocca CONT nello ZX80 riappare la lista del programma, premendolo un'altra volta compare la parola CONTINUE e premendo NEW LINE il programma prosegue dalla istruzione dopo lo STOP. E' cosi' andato perso il precedente contenuto dello schermo, ma non sono andati persi i risultati precedenti che sono rimasti in memoria. Dopo uno STOP si puo' anche proseguire con GOTO n, pero' anche in questo caso appena si tocca un qualunque tasto ricompare la lista, con GOTO n si prosegue, ma vanno persi i precedenti contenuti del video.

Nello ZX81 e nello ZX80-Nuova ROM non si ha questo inconveniente che la pressione di un tasto fa riapparire la lista del programma. Per avere la lista del programma si deve usare il comando LIST.

Se mentre e' presente un programma in memoria si eseguono delle istruzioni in modo immediato, il programma non viene disturbato. Naturalmente se si usano delle istruzioni di assegnazione (LET) possono essere modificati i valori di variabili gia' utilizzate dal programma andando ad influenzare i risultati finali.
E' molto comodo usare delle PRINT in modo immediato agli STOP programmati nei programmi in fase di prova. Anzi, se un programma e' complicato, e' buona norma inserire un certo numero di STOP nei punti chiave e poi toglierli dopo la prova definitiva.

Durante le prove dei programmi possono verificarsi delle situazioni di emergenza; per esempio, avere un ciclo dal quale non si esce, come il seguente:

```
10 INPUT N      chiede un numero
20 PRINT N      stampa il numero
30 GOTO 10      torna alla linea 10
```

In questo caso, se si risponde con lettere invece che con cifre alla richiesta di dato il calcolatore si ferma segnalando errore.

Se invece il programma ha un ciclo errato dal quale non si esce piu', ma non ci sono istruzioni INPUT, per uscire si puo' usare il tasto BREAK. Questo interompe l'esecuzione del programma e provoca uno STOP forzato. Si puo' continuare l'esecuzione con CONT.

Il tasto BREAK non ha effetto se il calcolatore e' in attesa di INPUT, mentre ha effetto durante l'uso del nastro e della stampante. Il sistema al termine dell'esecuzione di una linea di programma esamina il buffer della tastiera per vedere se e' stato premuto un tasto; se questo e' BREAK il programma si interrompe.

In caso di emergenza totale, cioe' quando non si sa piu' cosa fare, si puo' spegnere il calcolatore. Si ricordi pero' che spegnendo il calcolatore si perde il contenuto della RAM.

6.4. MEMORIZZAZIONE DI UN PROGRAMMA SU NASTRO

Per il calcolatore ZX80 si deve procedere cosi':

- mettere il registratore in grado di registrare la voce con i collegamenti al calcolatore staccati;
- avviare il registratore;
- registrare parlando il nome del programma e fermare il registratore;
- inserire il collegamento MIC (o REC) tra calcolatore e registratore;
- riavviare il registratore;
- premere subito sulla tastiera SAVE e NEW LINE.

A questo punto lo schermo si oscura, si vedono comparire delle righe orizzontali ed alla fine ricompare la lista del programma; attendere 10 secondi e fermare il registratore.

Se il registratore ha il controllo del livello di registrazione, e' bene assicurarsi tramite l'apposito indicatore che il segnale sia registrato ad un livello sufficientemente alto.

Assicurarsi che il registratore sia in buone condizioni di funzionamento.

Su uno stesso nastro possono essere registrati piu'

programmi, ma si deve fare attenzione a non sovrapporli. La ricerca va poi fatta in base al nome registrato a voce.

Per il calcolatore ZX81 e ZX80-Nuova ROM si deve procedere cosi':

. inserire il collegamento MIC (o REC) tra calcolatore e registratore;
. avviare il registratore;
. scrivere subito sulla tastiera: SAVE "nome-programma" e premere NEW LINE.

A questo punto succedono le stesse cose dette sopra. Quando compare 0/0 in basso a sinistra, attendere 10 secondi e fermare il registratore.
Valgono le stesse osservazioni fatte sopra riguardo al registratore.

Su uno stesso nastro possono essere registrati piu' programmi, senza sovrapporli; la ricerca viene fatta in base al nome del programma registrato prima del programma stesso.

6.5. CARICAMENTO DI UN PROGRAMMA DA NASTRO

Per il calcolatore ZX80 procedere cosi':

. staccare i collegamenti calcolatore registratore;
. cercare sul nastro con l'audio il nome del programma;
. dopo la frase si sente un BRRR... e poi silenzio; fermare il registratore appena inizia il silenzio;
. inserire il collegamento EAR (o MONITOR) tra registratore e calcolatore;
. riavviare il nastro e premere subito LOAD e poi NEW LINE;
. lo schermo diventa grigio e poi appare la lista del programma;
. fermare il registratore.

Tenere basso il volume del registratore in fase di ascolto, ma alzarlo in fase di caricamento programma.

Per il calcolatore ZX81 e ZX80-Nuova ROM procedere cosi':

. inserire il collegamento EAR (o MONITOR) tra calcolatore e registratore;
. scrivere subito:
. o LOAD "" ("" significa stringa nulla); ed in questo caso viene caricato il primo programma incontrato sul nastro;

. o LOAD "nome-programma"; ed in questo caso viene cercato e caricato il programma avente il nome richiesto.

Il nome di un programma non puo' superare 127 caratteri.
Il volume del registratore deve essere mantenuto sufficientemente alto.

CAPITOLO 7

U T I L I Z Z O D E L L A M E M O R I A

7.1. LA MEMORIA RAM E LA MEMORIA ROM

La memoria e' formata da elementi a due stati; se uno stato viene rappresentato da 0 e l'altro da 1 si puo' ragionare in termini di aritmetica binaria.
La memoria dei calcolatori SINCLAIR e' formata da questi elementi raggruppati 8 a 8. Il gruppo di 8 elementi prende il nome di BYTE, ed ogni elemento prende il nome di BIT da Binary digIT.

La grandezza della memoria si misura in byte. Il calcolatore standard ha la memoria RAM di 1K byte. K ha il valore convenzionale di 1024, quindi la memoria RAM standard e' di 1024 byte, cioe' 1024 gruppi di 8 bit.

Ogni byte e' indirizzabile singolarmente. La memoria RAM comincia all'indirizzo 16384, e, se e' di 1 solo K, termina all'indirizzo 17407. Se si aggiunge la memoria addiziomnale di 3K, gli indirizzi della RAM vanno da 16384 a 20479. Se, invece si aggiunge la memoria addizionale di 16K, ed allora viene escluso il K standard, gli indirizzi vanno da 16384 a 32767.
Ogni byte puo' contenere un numero che al massimo e' formato da 8 cifre 1 consecutive, tale numero corrisponde a 255 nel sistema di numerazione decimale.

Un qualunque numero decimale, per esempio: 7645, si puo' scrivere cosi':

7645 = 7*10**3 + 6*10**2 + 4*10**1 + 5*10**0
cioe':
7645 = 7000 + 600 + 40 + 5

Analogamente se si considera il numero del sistema binario: 11111111, si vede che esso si puo' scrivere:

11111111 = 1*2**7 + 1*2**6 + 1*2**5 + 1*2**4 + 1*2**3 + 1*2**2 + 1*2**1 + 1*2**0

Per non confondere tra loro numeri appartenenti a sistemi di numerazione diversi, essi si possono scrivere tra parentesi riportando in basso a destra la base del sistema di numerazione usato.

Nell'aritmetica binaria si fanno regolarmente i calcoli; le regole base sono:

1+1=0 con riporto di 1 e 1+0=1.

Dal momento che i numeri binari sono difficilmente leggibili, si usa interpretarli come appartenenti al sistema esadecimale, di base 16, raggruppando i bit 4 a 4, infatti 2 elevato a 4 da' 16.
In tale modo un byte risulta formato da 2 cifre esadecimali, di piu' facile lettura. Nel sistema esadecimale sono necessari 16 simboli diversi per rappresentare i numeri; era ovvio scegliere le cifre da 0 a 9 e poi le prime 6 lettere dell'alfabeto da A ad F. Cosi' A corrisponde a 10 decimale, B a 11, C a 12, D a 13, E a 14 ed F a 15. Il byte che contiene 255 in decimale puo' essere letto come FF in base 16 e come 11111111 in base 2.

Nei calcolatori SINCLAIR i numeri interi sono memorizzati in due byte consecutivi, con le cifre meno significative nel primo byte e le piu' significative nel secondo. L'indirizzo del numero e' pero' quello del primo byte, avente indirizzo pari. Cosi', per esempio, se all'indirizzo 16000 e' memorizzato il numero 3427 si ha:

- nel byte 16000 la parte meno significativa e cioe' 0110 0011;
- nel byte 16001 la parte piu' significativa e cioe' 0000 1101;

leggendoli in esadecimale il contenuto di 16000 e' 63 e quello di 16001 e' 0D.

Usando la funzione Basic PEEK per leggere i 2 byte per ricostruire il numero che questi rappresentano, si deve procedere cosi':

```
10 LET A = PEEK(16000)
20 LET B = PEEK(16001)
30 LET N = B * 256 + A
40 PRINT N
```

I numeri interi positivi hanno il primo bit del byte piu' significativo a zero. I numeri interi negativi sono memorizzati nella forma del complemento a 2 e quindi hanno il primo bit del byte piu' significativo a 1.

I numeri decimali (notazione esponenziale) sono sempre registrati con il valore assoluto della mantissa; il primo bit del byte piu' alto e' a 0 per i numeri positivi e ad 1 per i numeri negativi.

La memoria ROM dello ZX80 e' di 4K ed occupa i byte da 0 a 4095; la Nuova ROM e la ROM dello ZX81 e' di 8K ed occupa i byte da 0 a 8191. Dal momento che la RAM inizia al byte 16384 si hanno ancora, nel primo caso 12K e nel secondo 8K disponibili per future espansioni.

Nella memoria ROM sono stabilmente memorizzati i programmi che costituiscono il Sistema Operativo e l'Interprete Basic. L'utente non puo' scrivere nella ROM e non possono scrivere nella ROM neanche i programmi di sistema. Per questa ragione e' necessario che una parte della memoria RAM sia a disposizione del sistema per la memorizzazione delle variabili necessarie alla gestione.

7.2. LA PAGINA ZERO DELLA RAM

Si chiama "pagina zero", perche' e' la prima parte della RAM; i suoi indirizzi iniziano a 16384.

Si riportano separatamente le mappe della memoria per le due configurazioni dei calcolatori. Nella Appendice B sono descritte le variabili della pagina zero.

MAPPA MEMORIA ZX80

Utilizzo zona	Commento
Variabili del sistema	Indirizzo fisso di inizio 16384.
Programma utente	Indirizzo fisso di inizio 16424.
Area variabili programma	Questo indirizzo e' contenuto nel puntatore VARS (16392-16393).
Byte chiusura zona va= riabili	Questo byte contiene 128.
Area di lavoro	Questo indirizzo e' contenuto nel puntatore E-Line (16394-16395).
Area di memoria dedicata allo schermo	L'indirizzo di inizio di questa zona e' contenuto nel puntatore D-File (16396-16397); l'indirizzo della fine della zona sta nel pun=

	tatore DF-END (16400-16401). Nel puntatore DF-EA (16398-16399) si ha invece l'indirizzo di inizio della parte bassa dello schermo, quella dove si formano i comandi.
Area di memoria residua	L'indirizzo finale di questa zona viene indicato come RAMTOP.
Area STACK	Questa zona inizia all'ultimo in= dirizzo 17407 e si incrementa per indirizzi decrescenti.Il suo primo indirizzo disponibile e' puntato da SP, registro dello ZX80.

La prima zona "variabili del sistema" e' formata da 40 byte, si veda l'Appendice B per la descrizione dei contenuti. A questa zona appartengono i diversi puntatori citati nella tabella di cui sopra. Il metodo dei puntatori alle diverse zone della memoria consente di sfruttare al massimo, a seconda delle necessita', la capacita' della memoria. E' evidente che i puntatori devono avere una localizzazione fissa.

La zona programma inizia sempre all'indirizzo 16424 e termina prima della zona variabili. Subito dopo inizia la zona variabili, il cui indirizzo (variabile in dipendenza della lunghezza del programma) e' contenuto nel puntatore VARS. La zona delle variabili e' chiusa da un byte contenente 128 in decimale, 80 in esadecimale e 10000000 in binario.

La zona di lavoro, il cui indirizzo di inizio si trova in E-Line viene usata dal sistema per diverse esigenze. La zona di memoria destinata al video non ha dimensioni fisse, cioe' non e' "mappata in memoria", essa ha al minimo dimensione di 25 byte contenenti il carattere NEW LINE (76 in base sedici). Il primo e l'ultimo byte sono sempre a NEW LINE, tra questi vi sono 24 linee da 0 a 32 caratteri ciascuna. Tale zona prende anche il nome di "display file".

Il registro SP del sistema punta all'area STACK, che inizia dal fondo della memoria ed e' gestita per indirizzi decrescenti. Tale area viene usata in base al principio che l'ultimo dato depositato e' il primo ad uscire e serve come memoria di lavoro per quelle operazioni per le quali questo tipo di gestione ha un significato logico.

Se un programma e' troppo lungo, la zona dedicata al video diminuisce e si nota che lo schermo non puo' essere utilizzato tutto. Se si arriva ad occupare anche la zona dedicata alla STACK area si ha una segnalazione di errore.

MAPPA MEMORIA ZX81 E ZX80-NUOVA ROM

Utilizzo zona	Commento
Variabili del sistema	Indirizzo fisso di inizio 16384.
Programma	Indirizzo fisso di inizio 16509.
Memoria di schermo (Display File)	Puntatore all'inizio D-FILE (16396-16397).
Area Variabili del Programma	Puntatore all'inizio VARS (16400-16401).
Byte che chiude la zona Variabili	Contenuto del puntatore E-LINE. meno uno. Il contenuto del byte e' 80 esadecimale (128 dec.).
Area per la linea da scrivere + Area di lavoro	Puntatore all'inizio E-LINE (16404-16405).
Area Stack per il calcolatore	Puntatore all'inizio STKBOT (16410-16411).
Area libera	Puntatore all'inizio STKEND (16412-16413).
Area Stack per il microprocessore	Puntatore registro SP.
Area Stack per GOSUB	Puntatore all'inizio ERR-SP (16386-16387).
Area per programmi in linguaggio macchina (USR)	Puntatore all'inizio RAMTOP. In= dica il primo byte libero dopo il programma BASIC (16388-16389).

I primi 125 byte della memoria RAM sono utilizzati dal sistema, nell'Appendice B e' riportata la descrizione dei contenuti.

Al momento dell'accensione del calcolatore RAMTOP contiene l'indirizzo del primo byte non esistente nella memoria. Se si vogliono introdurre delle routine in linguaggio macchina, accessibili con il comando USR, si puo' modificare con una POKE il contenuto di RAMTOP e caricare le routine a partire dall'indirizzo contenuto in RAMTOP. Il vantaggio di questa procedura e' che il comando NEW non tocca le posizioni di memoria oltre il contenuto di RAMTOP, lo svantaggio e' che

il contenuto di questo ultimo pezzo di memoria non viene salvato sul nastro quando si memorizza il programma in BASIC con il comando SAVE. Inoltre il programma BASIC non interferisce con la zona di memoria che inizia all'indirizzo contenuto in RAMTOP.

La memoria di schermo inizia dopo il programma all'indirizzo contenuto in D-FILE. La memoria di schermo puo' contenere 24 linee, ciascuna di 32 caratteri + il carattere NEW LINE. A seconda delle dimensioni della RAM del calcolatore il sistema riserva per lo schermo una zona completa, cioe' di 24*33 caratteri, o una zona di dimensioni minori. Se, tenendo conto del valore contenuto in RAMTOP, si ha a disposizione poca memoria il sistema assegna alla memoria di schermo le dimensioni minime di 25 caratteri ed essi alla partenza del sistema o per effetto del comando CLS sono 25 caratteri NEW LINE. Inserendo la RAM aggiuntiva di 16K la memoria di schermo e' completamente mappata.

E-LINE contiene l'indirizzo di inizio della parte di memoria dove:

- si sta scrivendo: un comando, una linea di programma o un dato di INPUT
- e' disponibile una parte di memoria per lavorare.

STKBOT contiene l'indirizzo di inizio dell'area usata per i calcoli, mentre il registro SP punta all'area stack usata dal microprocessore ZX80.

7.3. COME SONO MEMORIZZATI I PROGRAMMI

Nello ZX80 le linee di programma sono memorizzate cosi':

Primo byte	Byte piu' significativo del numero di linea.
Secondo byte	Byte meno significativo del numero di linea.
Byte seguenti	Testo della linea.
Ultimo byte	NEW LINE (76 esadecimale, 118 decimale).

Si noti che il numero della linea e' memorizzato ponendo a sinistra il byte piu' significativo ed a destra il meno significativo, in modo contrario al comportamento abituale dello ZX80. Dato che sono ammessi numeri di linea da 1 a 9999, si vede subito che il byte piu' significativo di tali numeri ha i primi 2 bit di sinistra uguali a zero. Come si vedra' nei prossimi paragrafi, le variabili sono rappresentate in modo da non avere mai i primi 2 bit a zero; quindi l'incontrare dopo il carattere NEW LINE, che chiude sempre una istruzione, un byte con in primi due bit non uguali a 00, segnala che il programma e' terminato. Comunque la zona inizio variabili e' rilevabile dal puntatore VARS.

Nel testo della linea le parole chiave ed i simboli del linguaggio occupano sempre un solo byte ciascuno, le costanti ed i nomi simbolici inventati dal programmatore sono registrati carattere per carattere.

Nello ZX81 e nello ZX80-Nuova ROM le linee di programma sono memorizzate cosi':

Primo byte	Byte piu' significativo del numero di linea.
Secondo byte	Byte meno significativo del numero di linea.
Terzo e quarto byte	Lunghezza in byte dell'istruzione + 1 per il byte con NEW LINE.
Bytes successivi	Istruzione.
Ultimo byte	NEW LINE corrispondente a 01110110 in binario (76 in esadecimale e 118 in decimale).

7.4. COME SONO MEMORIZZATI I DATI

Nello ZX80 i dati sono memorizzati secondo le modalita' descritte nel seguito.

MEMORIZZAZIONE DELLE VARIABILI

Le variabili hanno tutte nomi simbolici che iniziano con una lettera, i codici rappresentativi delle lettere vanno da 38 a 63 in decimale e quindi da 26 a 3F in esadecimale. Tutte le lettere hanno in conseguenza un codice di 6 bit ed il primo bit e' sempre 1. Come si vede dagli schemi riportati, il sistema gioca sui primi bit delle lettere aggiungendone altri, i primi due, ed eventualmente azzerando il terzo, per distinguere tra loro i diversi tipi di variabili che tratta.

VARIABILE NUMERICA CON NOME DI UNA SOLA LETTERA

Primo byte	011 + altri 5 bit codice lettera.
Secondo byte	Byte meno significativo numero.
Terzo byte	Byte piu' significativo numero.

Per ogni variabile di questo tipo sono occupati 3 byte. Le variabili numeriche dello ZX80 riguardano solo numeri interi in valore assoluto minori o uguali a 32767.

VARIABILE NUMERICA CON NOME LUNGO

Primo byte	010 + altri 5 bit codice prima lettera.
Secondo byte	00 + secondo carattere nome.
Byte seguenti	00 + altri caratteri nome.
Byte ultimo caratt. nome	10 + ultimo carattere nome.
1 byte	Byte meno significativo numero.
1 byte	Byte piu' significativo numero.

VARIABILE STRINGA

Primo byte	100 + altri 5 bit codice lettera nome.
Byte seguenti	I caratteri della stringa in sequenza.
Ultimo byte	Codice del carattere apici per chiudere (00000001 binario).

VARIABILE NUMERICA CON INDICE

Primo byte	101 + altri 5 bit codice lettera nome.
Secondo byte	Valore dell'indice usato nella DIM, quindi numero degli elementi - 1.
2 byte	Per il primo elemento, di indice 0, nell'ordine: meno significativo e piu' significativo.
Coppie 2 byte	Per gli elementi successivi.

VARIABILE DI CONTROLLO PER I CICLI FOR/NEXT

Primo byte	111 + altri 5 bit codice lettera nome.
2 byte	Valore iniziale variabile controllo.
2 byte	Valore limite dopo il TO.
2 byte	Numero della linea dell'istruzione FOR aumentato di 1 (se questo numero di linea non esiste nel programma, il sistema cerca quella di numero immediatamente superiore).

Nello ZX81 e nello ZX80-Nuova ROM le variabili sono memorizzate come viene descritto nel seguito.

Le variabili del BASIC hanno tutte nomi simbolici che iniziano con una lettera, i codici ASCII delle lettere sono compresi tra 38 e 63 (tra 26 e 3F in esadecimale) e quindi hanno un codice con solo 6 bit significativi, il primo dei quali a sinistra e' sempre 1. Come si puo' osservare negli schemi che seguono il sistema gioca sui primi bit del primo carattere del nome per distinguere tra loro i diversi tipi di variabili ed inoltre, in alcuni casi, anche sui primi bit

dei caratteri successivi.

VARIABILE NUMERICA CON NOME DI UNA SOLA LETTERA

Primo byte	0 1 1 + altri 5 bit codice lettera.
Secondo byte	Caratteristica del numero (esponente).
4 byte	Mantissa del numero con segno.

Per ogni variabile di questo tipo sono occupati 6 byte.

VARIABILE NUMERICA CON NOME LUNGO

Primo byte	1 0 1 + altri 5 bit codice prima lettera.
Secondo byte	0 0 + secondo carattere nome.
.	
.	
.	
Ultimo byte nome	1 0 + ultimo carattere nome.
5 byte	Valore del numero (1 byte per esponente + 4 byte per mantissa).

Per ogni variabile di questo tipo sono occupati 5 byte + 1 byte per ogni carattere del nome.

VARIABILI NUMERICHE CON INDICE

Primo byte	1 0 0 + ultimi 5 bit codice lettera avendo sostituito il primo bit 1 dello stesso codice con 0.
Secondo e terzo	Numero byte occupati = (5 * numero ele= menti + (2 * numero-dimensioni) + 1).
Quarto byte	Numero delle dimensioni.
2 byte per ogni dimens.	Valore della dimensione. Si ha una coppia di byte per ogni dimensione.
5 byte per ogni elem.	Valore dell'elemento: esponente e mantissa.

L'ordine degli elementi e' quello che si ottiene facendo variare piu' rapidamente gli indici piu' a destra e muovendosi verso sinistra. Esempi:

A(2,3)   viene disposto in memoria cosi':

A(1,1),A(1,2),A(1,3),A(2,1),A(2,2),A(2,3)

B(2,3,4) viene disposto in memoria cosi':

B(1,1,1),B(1,1,2),B(1,1,3),B(1,1,4),B(1,2,1),B(1,2,2),.....
.....................B(2,3,3),B(2,3,4)

VARIABILI DI CONTROLLO PER I CICLI FOR-NEXT

Queste variabili possono avere il nome formato da una sola lettera.

Primo byte	1 1 1 + ultimi 5 bit codice lettera.
5 byte	Valore iniziale variabile di controllo.
5 byte	Valore finale variabile di controllo.
5 byte	Valore dello STEP.
2 byte	Numero di linea della linea del FOR + 1 (se tale linea non esiste il sistema cerca quella immediatamente superiore).

VARIABILI STRINGA

Queste variabili possono avere il nome formato da una sola lettera + il carattere $.

Primo byte	0 1 0 + ultimi 5 bit del codice lettera avendo sostituito il primo bit del codice con 0.
Secondo e terzo byte	Numero dei caratteri della stringa, massi mo 32767. Tale numero viene limitato solo dalla disponibilita' di memoria.
Byte successivi	Testo della stringa. La stringa puo' essere vuota.

VARIABILI STRINGA CON INDICE

Queste variabili possono avere il nome formato da una sola lettera + il carattere $. Il numero delle dimensioni e' a piacere, ma ogni elemento deve avere la stessa dimensione.

Primo byte	1 1 0 + ultimi 5 bit del codice lettera avendo sostituito il primo bit del codice con 0.
Secondo e terzo byte	Numero byte occupati = (numero elementi * lunghezza elementi) + 1 + (2 * numero-di= mensioni) + 2.
Quarto byte	Numero dimensioni + 1.
2 byte per ogni dimens.	Valore della dimensione. Si ha una coppia di byte per ogni dimensione.
2 byte	Lunghezza in caratteri di ogni elemento.
Numero byte	Elementi uno dopo l'altro in ordine di

necessario per ogni elemento	indice facendo variare piu' rapidamente l'indice piu' a destra.

7.5. COME SONO MEMORIZZATI I CARATTERI PER IL VIDEO

Nella memoria ROM sono memorizzati tutti i caratteri stampabili dedicando ad ogni carattere 8 byte, cioe' ogni carattere e' rappresentato in una matrice di punti 8 per 8. Il carattere e' letteralmente disegnato usando i bit 1 in un campo tutto di bit 0. Vediamo il disegno della lettera A:

Primo byte	0 0 0 0 0 0 0 0	. . . .
Secondo byte	0 0 1 1 1 1 0 0	. . * * * * . .
Terzo byte	0 1 0 0 0 0 1 0	. * . . . . * .
Quarto byte	0 1 0 0 0 0 1 0	. * . . . . * .
Quinto byte	0 1 1 1 1 1 1 0	. * * * * * * .
Sesto byte	0 1 0 0 0 0 1 0	. * . . . . * .
Settimo byte	0 1 0 0 0 0 1 0	. * . . . . * .
Ottavo byte	0 0 0 0 0 0 0 0	. . . .

Dato che riferendosi a 0 e 1 non si vede bene il carattere si e' riportato vicino un disegno ottenuto sostituendo allo zero il punto e all'uno l'asterisco.

Quando il carattere viene stampato il sistema, usando una routine che fa parte del Sistema Operativo e si trova in ROM, riporta sul video proprio un punto (pixel) al posto dei bit 1 presenti nella matrice del carattere.

Nello ZX80 la mappa dei caratteri inizia all'indirizzo 3584, nello ZX81 e nello ZX80-Nuova ROM essa inizia all'indirizzo 7680. Spostandosi nella mappa con passo 8, 8 byte per volta, si trovano tutti i caratteri. Per trovare la rappresentazione di un carattere di codice X, chiamando B l'indirizzo di inizio della mappa dei caratteri, e usando un indice I che parte da 0 e arriva a 7, si procede cosi':

```
Indirizzo primo   byte (I=0) =  B + X * 8 + I
Indirizzo secondo byte (I=1) =  B + X * 8 + I
.....
Indirizzo ottavo  byte (I=7) =  B + X * 8 + I
```

La tabella dei caratteri occupa 512 byte e quindi (512/8=64) puo' contenere solo 64 caratteri; questi sono i 64 caratteri stampabili, il cui codice va da 0 a 63. I caratteri in campo inverso si ottengono invertendo il significato degli zeri e degli uno; il loro codice e' uguale a quello del carattere diretto aumentato di 128.

Si puo' usare la mappa dei caratteri per ottenere sul video dei caratteri ingranditi. Si puo' cioe' sfruttare la

rappresentazione di ogni carattere come maschera per andare a stampare, per esempio, lo spazio in campo inverso, dove nella maschera compare 1 e lo spazio dove compare 0. In tale modo si ottiene un ingrandimento di 8 volte del carattere. Se si vuole ingrandire di piu' si puo' anche farlo, ma esiste una limitazione dovuta alle dimensioni del video.

Nel Capitolo 9 sono riportati dei programmi che ingradiscono i caratteri.

7.6. ALCUNI CONSIGLI PER PROGRAMMARE BENE

Se si vuole programmare in modo ottimale un calcolatore relativamente piccolo come il SINCLAIR, si devono avere presenti due aspetti del problema; il primo riguarda l'occupazione della memoria, il secondo la velocita' esecutiva dei programmi. Le considerazioni da fare dipendono anche dalla memoria disponibile. Se si ha 1 solo K di memoria, e' evidente che la cosa piu' importante e' risparmiarla anche a scapito della velocita'.

Nel paragrafo 7.3 viene descritto come sono memorizzate le linee di programma e nel paragrafo 7.4. viene descritta la reale occupazione di memoria da parte dei dati nei due calcolatori. Si possono fare alcune considerazioni.

CALCOLATORE ZX80

Nello ZX80, che tratta solo numeri interi, questi occupano relativamente poco spazio, 3 byte, se il nome e' di una sola lettera. Analogamente le variabili intere con indice occupano 2 byte per elemento, piu' 1 byte per il numero degli elementi diminuito di 1, piu' 1 byte per il nome. Le stringhe invece occupano tanti byte quanti sono i caratteri piu' 2 (1 per il nome ed 1 per la chiusura della stringa, infatti non c'e' il contatore per il numero degli elementi). Da quanto detto si deduce che conviene tenere memorizzati i numeri in variabili numeriche; infatti un numero di 5 cifre trasformato in stringa occupa 7 byte contro i 3 necessari per il numero.

Si deve tener presente che le stringhe vengono definite quando ricevono una assegnazione di contenuto e l'occupazione di memoria dipende dal numero dei caratteri. Se in un programma si ha una istruzione del tipo:

```
10 INPUT A$
......
```

e si torna piu' volte a questa stessa istruzione, ogni volta

che A$ riceve un contenuto essa cambia di posto in memoria, anche se non cambia il numero dei caratteri. Ogni volta che la stringa cambia di posto il buco lasciato libero viene rioccupato spostando in su tutte le altre variabili e questo naturalmente rallenta i tempi di esecuzione. Si provi il seguente programma:

```
 10 LET A$ = "TRE"
 20 LET B$ = "SEC"
 30 PRINT "SCRIVI 3 CARATTERI"
 40 INPUT C$
 50 LET N = 35
 60 LET D$ = "TAPPO"
 70 GO SUB 200
 80 PRINT "SCRIVI 4 CARATTERI"
 90 INPUT C$
100 LET N = 36
110 GOSUB 200
120 STOP
200 LET M = 256 * PEEK(16393) + PEEK(16392)
210 LET N = M + N
220 FOR K = M TO N
230 PRINT PEEK(K);" ";
240 NEXT K
245 PRINT
250 RETURN
```

si vedra' che la stringa C$ viene creata una seconda volta alla linea 90, essa e' anche piu' lunga della precedente. I contenuti del video, se si risponde "ABC" alla prima richiesta di INPUT e "ABCD" alla seconda, con la necessaria interpretazione sono:

```
134  57  55  42   1
A$   T   R   E    "

135  56  42  40   1
B$   S   E   C    "

136  38  39  40   1
C$   A   B   C    "

115  31  65
N    (N + M) corrispondente al numero 35+16636
             16636 e' il contenuto del puntatore VARS

137  57  38  53  53  52   1
D$   T   A   P   P   O    "

114 252  64
M   (M) corrispondente al numero 16636
```

```
240  25  65   31  65   221   0
K    valore  limite  numero linea
     attuale   K     della FOR + 1
       K     (N+M)

128
fine zona variabili

134  57  55  42   1
A$   T   R   E    "

135  56  42  40   1
B$   S   E   C    "

115  32  65
N    (N+M) corrisponde al numero 36+16636
           16636 e' il contenuto di VARS

137  57  38  53  53  52   1
D$   T   A   P   P   O    "

114 252 64
M   (M) corrispondente al numero 16636

240  20  65   32  65   221   0
K    valore  limite  numero linea
     attuale   K     della FOR + 1
       K     (N+M)

136  38  39  40  41   1
C$   A   B   C   D    "

128
fine zona variabili
```

Come si puo' vedere la variabile C$ ha cambiato posto, cioe' e' stata cancellata la precedente variabile C$, tutte le altre variabili sono state spostate all'indietro e la nuova C$ e' stata messa in coda. Le variabili numeriche hanno invece conservato la loro posizione rispetto alle altre. Se provate a far girare il precedente programma di nuovo e rispondete alla richiesta di 4 caratteri ancora con 3, vedrete che la variabile C$ cambia ancora di posto, questo significa che le stringhe vengono sempre cancellate e riscritte anche se mantengono lo stesso numero di caratteri.

Per rendersi conto dell'occupazione di spazio da parte del programma si puo' fare la prova seguente, dopo aver premuto NEW e NEW LINE:

```
10 LET A = 1257
```

```
20 LET B = A
30 FOR K = 16424 TO 16474
40 PRINT PEEK(K);" ";
50 NEXT K
```

poi dare RUN; si vedranno sul video i contenuti dei primi 51 byte della memoria. Essi, con la relativa interpretazione, sono:

```
 0    10   240   38   227   29   30   33   35   118
 numero    LET   A     =    1    2    5    7    NEW LINE
 linea

 0    20   240   39   227   38   118
 numero    LET   B     =    A    NEW LINE
 linea

 0    30   235   48   227   29   34   32   30   32   214   29
 numero    FOR   K     =    1    6    4    2    4    TO    1
 linea

34    32    35   32   118
6     4     7    4    NEW LINE

 0    40   244   53    42    42   48   218   48   217   215   1
 numero   PRINT  P     E     E    K    (     K    )     ;     "
 linea
 0     1   215  118
spazio "    ;   NEW LINE

 0
spazio per segnalare la fine del programma.
```

Come si vede la seconda istruzione (LET B = A) occupa meno spazio della prima (LET A = 1257), per questa ragione conviene definire le costanti una sola volta come variabili e poi usare le corrispondenti variabili nel corso del programma.

Nei precedenti programmi esemplificativi si e' usato PRINT PEEK(K) e non PRINT PEEK(CHR$(K)) perche' alcuni caratteri ASCII non sono stampabili e quindi e' meglio riferirsi al codice numerico.

Possiamo iniziare facendo girare sul calcolatore con il nuovo Basic i due programmi discussi precedentemente per lo ZX80, dopo aver fatto le necessarie modifiche.

Il primo programma e' diventato il seguente:

```
 10 LET A$ = "TRE"
 20 LET B$ = "SEC"
 30 PRINT "SCRIVI 3 CARATTERI"
 40 INPUT C$
 50 LET N = 56
 60 LET D$ = "TAPPO"
 70 GO SUB 200
 80 PRINT "SCRIVI 4 CARATTERI"
 90 INPUT C$
100 LET N = 57
110 GO SUB 200
120 STOP
200 LET M = 256*PEEK(16401) + PEEK(16400)
210 LET N = N + M
220 FOR K = M TO N
230 PRINT PEEK(K);" ";
240 NEXT K
245 PRINT
250 RETURN
```

infatti in questo caso le variabili numeriche sono piu' lunghe, la memorizzazione delle stringhe e' ottenuta in un altro modo ed i caratteri occupati diventano 56 nel primo caso e 57 nel secondo. Inoltre il puntatore VARS ha indirizzo 16400. I risultati ottenuti, con la relativa interpretazione, sono:

```
 70    3      0    57    55    42
 A$    num.        T     R     E
       caratt.

 71    3      0    56    42    40
 B$    num.        S     E     C
       caratt.

 72    3      0    38    39    40
 C$    num.        A     B     C
       caratt.

115  143     9  210     0     0
N    esp.   mantissa
```

```
 73    5      0    57    38    53   53   52
 D$    num.        T     A     P    P    o
       caratt.

114  143    9   98     0     0
M    esp.   mantissa

240  143    9  180     0     0  143    9  210    0    0
K    valore iniziale var.  K  valore limite per var. K

129    0    0     0     0  221    0
valore dello STEP          numero linea FOR + 1

128
fine zona variabili
```

Lasciamo al lettore l'interpretazione della seconda parte dei risultati. Anche in questo caso la variabile C$ e' stata spostata in memoria. L'occupazione di memoria da parte delle variabili numeriche e' un po' pesante. Il valore della caratteristica dei numeri (esponente), qui espresso come numero decimale si riferisce al numero dei bit da spostare a sinistra del punto decimale per ottenere il valore del numero, e che per l'esponente lo zero e' rappresentato dal numero 128.

Per fissare le stringhe in memoria si puo' dimensionarle senza attribuire loro indici, ma assegnando loro una lunghezza in caratteri. Per esempio: DIM A$(7) fissa in memoria la stringa A$ lunga 7 caratteri.

Per fare girare il secondo programma si deve modificare l'indirizzo del byte di inizio dei programmi che e' ora 16509. Il programma e' ora:

```
10 LET A = 1257
20 LET B = A
30 FOR K = 16509 TO 16589
40 PRINT PEEK(K);" ";
50 NEXT K
```

ed esso occupa piu' byte in memoria della versione precedente, infatti nel nuovo Basic le istruzioni occupano piu' memoria. I risultati, con la relativa interpretazione, sono:

```
 0   10   numero linea

14    0   lunghezza in byte istruzione

241   38   20   29   30   33   35
LET   A    =    1    2    5    7
```

```
126 139   29   32    0    0
    numero 1247 in floating-point

118
NEW LINE

 0   20  numero linea

 5    0  lunghezza in byte istruzione

241  39   20   38       118
LET  B    =    A        NEW LINE

 0   30  numero linea
27    0  lunghezza in byte istruzione

235   48   20   29   34   33   28   37
FOR   K    =    1    6    5    0    9

126 143   0  250    0    0
    numero 16509 in floating-point

223   29   34   33   36   37
TO    1    6    5    8    9

126 143   1  136    0    0
    numero 16589 in floating-point

118
NEW LINE

 0   40  numero linea

11    0  lunghezza in byte istruzione

245    211   16   48   17   25  11   0  11   25     118
PRINT  PEEK  (    K    )    ;   "  spazio "   ;    NEW LINE

 0   50  numero linea

 3    0  lunghezza in byte istruzione

243   48  118
NEXT  K   NEW LINE

118
NEW LINE  di fine programma
```

Come si vede, in questo caso l'occupazione di memoria che si ha incorporando direttamente nelle istruzioni dei numeri come costanti e' piuttosto pesante, infatti prima viene conservato il numero cifra per cifra e poi, dopo il codice

126 di inizio "literal", si ha il numero in floating-point. Il sistema si comporta cosi' per evitare di dover convertire ogni volta nel numero floating-point e quindi si guadagna in velocita' a scapito dell'occupazione della memoria.

In ogni programma si deve decidere cosa conviene fare; se una costante e' usata una sola volta vale la pena di lasciarla nella istruzione che la usa, se e' usata piu' volte conviene definirla a parte e poi richiamarla con il suo nome.

Ricordando che la condizione VERO corrisponde al valore 1 della variabile logica e che la condizione FALSO corrisponde al valore 0, potete avere a disposizione uno 0 o un 1 nel programma scrivendo:

```
LET  A = X = X
```

sempre che X sia una variabile gia' esistente nel programma, la precedente istruzione pone A=1; se invece scrivete:

```
LET A = NOT X = X
```

ottenete A=0.

Per valutare le differenze in tempi di esecuzione tra i diversi modi di scrivere un programma potete fare le seguenti prove:

```
PRO1:  100 LET A = 5
       110 FOR K = 1 TO 2000
       120 LET B = 8
       130 NEXT K
       140 STOP
```

```
PRO2:  100 LET A = 5
       110 FOR K = 1 TO 2000
       120 LET B = A
       130 NEXT K
       140 STOP
```

nei due programmi esiste solo una differenza nella istruzione 120. La differenza del tempo di esecuzione delle due istruzioni viene moltiplicata per 2000 eseguendo il ciclo FOR. Se misurate il tempo di esecuzione tra il RUN e lo STOP vedrete una piccola differenza.

Provate poi di nuovo i due programmi sostituendo in PRO1 la linea 120 con la: 120 LET B = 1, e in PRO2 la linea 120 con la: 120 LET B = A = A e calcolate le differenze nei tempi di esecuzione.

Potete fare una ulteriore prova ponendo
in PRO1: 120 LET B = 0
e in PRO2: 120 LET B = NOT A = A.

Da quanto visto fino ad ora risulta che i numeri occupano molto spazio in memoria e che quindi puo' essere consigliabile trovare degli accorgimenti di programmazione

che aiutino a risparmiare, magari a scapito della velocita'. Supponiamo di avere bisogno di una tabella di dati numerici, contenente 10 elementi, e che i numeri siano al massimo di 3 cifre. Sara' necessario dimensionare la tabella e poi riempirla con i numeri:

```
 10 DIM T(10)
 20 LET T(1) = 123
 30 LET T(2) = 90
 .....
 .....
100 LET T(10) = 567
```

e questo pezzo di programma occupa parecchia memoria. Pero' si puo' procedere anche cosi':

```
10 DIM T(10)
20 LET A$ = "123090............567"
30 FOR K = 0 TO 9
40 LET T(K+1) = VAL A$(K*3+1 TO K*3+3)
50 NEXT K
60 LET A$ = ""
```

e con questo pezzo di programma si ottiene di caricare i numeri, preventivamente generati nella stringa A$, negli elementi della tabella. L'istruzione 60 distrugge la stringa oramai adoperata e libera la memoria occupata. Naturalmente per far girare il programma una seconda volta si deve ricaricarlo da nastro in memoria o ricostruire in modo immediato la stringa A$. Questo sistema funziona se si memorizzano i numeri nella stringa tutti con lo stesso numero di cifre.

Per valutare i tempi di esecuzione si possono modificare i programmi PRO1 e PRO2 in questo modo:

- . sostituire la linea 100 con: 100 LET A$ = "234"
- . sostituire in PRO1 la linea 120 con: 120 LET B=234
- . sostituire in PRO2 la linea 120 con: 120 LET B=VAL A$

e provare i due programmi valutando i tempi.

Nell'esempio appena visto, coloro che conoscono il comando DATA, presente in altre implementazioni del Basic, avranno ritrovato una simulazione del medesimo, con la limitazione di avere sistemato nella stringa A$ elementi tutti della stessa lunghezza. Questo inconveniente puo' essere superato aggiungendo un carattere delimitatore tra gli elementi memorizzati sotto forma di stringa e scrivendo un programma di caricamento dalla tabella piu' complicato del precedente, che analizzi la presenza del carattere separatore per decidere la fine di ogni elemento.

Per rendere piu' veloci i programmi e' buona norma sistemare i sottoprogrammi all'inizio del programma, infatti in presenza di un GOSUB il sistema ricerca dall'inizio del programma il numero di linea voluto. Il programma puo' inziare cosi':

10 GOTO 1000   e in 1000 inizia il programma principale

dopo la linea 10 vengono sistemati tutti i sottoprogrammi.

La tecnica dell'uso dei sottoprogrammi e' consigliabile sia per risparmiare memoria che per avere dei programmi facilmente leggibili. Naturalmente tutte le parti componenti un programma dovrebbero essere precedute da una bella serie di REM con tutti i commenti esplicativi necessari; pero' cosi' si consuma tanta memoria! Si dovra' arrivare ad un compromesso con la capacita' di memoria e scrivere le note a parte nella documentazione del programma.

Per risparmiare memoria si puo' evitare di mettere in un programma le linee di assegnazione dei valori iniziali alle variabili (LET.....) e, dopo aver scritto il programma, caricare in modo immediato le variabili con i loro contenuti iniziali. Subito dopo il programma deve essere memorizzato su nastro; in tale modo i valori iniziali delle variabili vanno a fare parte del programma. Si ha pero' l'inconveniente che questo programma non puo' essere mandato in esecuzione con RUN perche' verrebbero cancellati i contenuti delle variabili, ma deve essere mandato in esecuzione con GOTO N.

7.7. LA PRECISIONE NEI CALCOLI

Ogni calcolatore puo' trattare numeri di una limitata grandezza in dipendenza dalle sue caratteristiche. Lo ZX80 tratta solo numeri interi in valore assoluto minori di 32767. Lo ZX81 e lo ZX80-Nuova ROM possono trattare numeri interi o decimali in valore assoluto minori di 4294967295.

Le modalita' di stampa dei numeri possono mostrare meno cifre di quante realmente conservate in memoria.

Anche tenendo presente quanto detto, si possono avere delle sorprese nei calcoli, dato che i numeri non sono trattati come decimali, ma vengono convertiti in binario.

Si possono fare delle prove; per esempio introdurre un numero decimale in notazione decimale e lo stesso numero in notazione esponenziale, e poi usando la PEEK andare a vedere come e' stato memorizzato realmente. Esempio:

```
10 INPUT A
20 INPUT B
30 LET M = 256*PEEK(16401) + PEEK(16400)
40 FOR K = 1 TO 12
50 PRINT PEEK(M+K-1);" ";
60 NEXT K
```

In questo programma si leggono A e B e si deve rispondere dando per A un numero in notazione decimale e per B lo stesso numero in notazione esponenziale. M viene posta uguale all'indirizzo di inizio delle variabili (VARS) e con un ciclo vengono stampati i 12 byte delle variabili A e B.

Si riportano alcuni risultati ottenuti:

A	B	Contenuto dei 6 byte					
0.125		102	125	127	255	255	255
	125E-3	103	126	0	0	0	0
0.5		102	127	127	255	255	255
	5E-1	103	128	0	0	0	0
0.625		102	128	31	255	255	255
	625E-3	103	128	32	0	0	0
0.33		102	127	40	245	194	143
	33E-2	103	127	40	245	194	143
5		102	131	32	0	0	0
	5E0	103	131	32	0	0	0
45327		102	144	49	15	0	0
	0.45327E+5	103	144	49	15	0	1
4294967295		102	160	127	255	255	255
	42949.67295E+5	103	160	127	255	255	255
0.000375		102	117	68	155	165	226
	375E-6	103	117	68	155	165	227

E' evidente che se al programma precedente si aggiunge un controllo sull'uguaglianza di A e B in alcuni casi si otterrebbe la non uguaglianza.

Da quanto visto ora si deduce che sarebbe sempre consigliabile introdurre i numeri decimali in notazione esponenziale.

7.8. LA MEMORIA DI SCHERMO

Nel paragrafo 7.2 si e' visto come, tramite i puntatori si puo' risalire agli indirizzi della memoria di schermo. Nello ZX80 la memoria di schermo e' sempre di dimensioni variabili, anche se si aggiunge l'espansione RAM. Nello ZX81 e nello ZX80-Nuova ROM invece, se si aggiunge l'espansione di memoria da 16K, la memoria di schermo ha le dimensioni fisse di 793 byte (33x24+1) pero' si sposta nella memoria in dipendenza dalla lunghezza del programma.

Il programma che segue, riempie con lo spazio inverso (CHR$(128)) le prime due righe del video, poi legge dal puntatore D FILE l'indirizzo di inizio della memoria di schermo e dal puntatore VARS l'indirizzo di inizio della zona variabili; la differenza dei due indirizzi da' la lunghezza della memoria di schermo (cioe' 793). Il programma stampa questi due indirizzi. Il contenuto di una parte della memoria di schermo viene memorizzato in un vettore A e poi viene stampato il valore del codice. Si vede 118 per il NEW LINE iniziale, poi 32 volte 128, poi ancora 118 ed infine ancora 32 volte 128.

Nel caso specifico si poteva fare a meno di memorizzzare il contenuto della memoria di schermo in altra zona di memoria (vettore A), dato che lo schermo resta mezzo vuoto e non si rischia di cancellarlo. In altri casi questo metodo e' necessario perche' la memoria di schermo si modifica facilmente e si rischia di perdere i precedenti contenuti che si volevano analizzare.

```
 10 DIM A(67)
 20 FOR L = 1 TO 2
 30 FOR J = 1 TO 32
 40 PRINT CHR$(128);
 50 NEXTJ
 55 PRINT
 60 NEXT L
 70 GOSUB 100
 80 GOSUB 200
 90 STOP
100 LET M = 256*PEEK 16397 +PEEK 16396
110 LET N = 256*PEEK 16401 +PEEK 16400
113 PRINT M,N
115 LET L = 1
120 FOR K = M TO M + 65
130 LET A(L) = PEEK K
135 LET L=L+1
140 NEXTK
150 RETURN
200 FOR I = 1 TO 66
```

```
210 PRINT A(I);" ";
220 NEXT I
230 RETURN
```

Nel caso in questione, e cioe' quando la memoria di schermo e' completamente mappata in memoria, si possono fare delle POKE negli indirizzi della memoria di schermo e si vedono comparire i relativi caratteri. Potete provare, partendo dall'indirizzo M che vedete stampato sul video, a mettere in diversi punti dello schermo dei caratteri usando le POKE.

Qualora voleste fare lo stesso tipo di prova con il calcolatore senza l'espansione RAM avreste delle sorprese, cioe' non potete fare delle POKE nella memoria di schermo se essa non e' mappata in memoria.

CAPITOLO 8

IL LINGUAGGIO MACCHINA

8.1. IL LINGUAGGIO DEL CALCOLATORE

Il linguaggio del calcolatore e' il linguaggio macchina. Nella Appendice F sono riportati: nella prima colonna le istruzioni in linguaggio simbolico Assembler del microprocessore Z80, nella seconda il corrispondente codice macchina espresso in esadecimale, nella terza il corrispondente valore decimale e nella quarta un breve commento. La prima colonna esprime in forma mnemonica le istruzioni per il calcolatore. Si potrebbe scrivere un programma usando le istruzioni simboliche assembler, ma poi sarebbe necessario un programma assemblatore per tradurle in codice macchina prima di poterle eseguire.

Per i calcolatori Sinclair noi possiamo scrivere programmi in linguaggio macchina, ma dobbiamo codificarli in codice macchina e caricarli nella memoria del calcolatore o in codice decimale o in codice esadecimale, come vedremo nel prossimo paragrafo. Non disponiamo infatti di un programma assemblatore.

Non possiamo in questa sede descrivere tutte le istruzioni disponibili; esse sono listate nella Appendice F con un commento sicuramente non esauriente. Coloro che conoscono gia' altri linguaggi di tipo Assembler potranno solo con pochi riferimenti riuscire a scrivere piccoli programmi. Coloro che non si sono mai occupati di linguaggi di questo tipo dovranno documentarsi su altri testi piu' completi. "Il NANOBOOK Z-80 - Vol.1 - Tecniche di programmazione", pubblicato dal Gruppo Editoriale Jackson, puo' essere utile allo scopo.

Si ricordi che in linguaggio macchina si devono scrivere tutte le istruzioni elementari per ottenere una qualunque operazione, i calcoli si svolgono in particolari registri chiamati accumulatori. Le istruzioni sono di lunghezza variabile e possono occupare da uno a quattro byte.

Riportiamo un piccolo esempio di sottoprogramma che viene mandato in esecuzione da un programma Basic. Si tratta di 6

istruzioni che svolgono questo calcolo:

. viene caricato nell'accumulatore A un dato numerico, e precisamente quello che si trova nel secondo byte della prima istruzione;

. viene incrementato di 1 per due volte l'accumulatore A, e quindi in A si trova il numero precedentemente caricato + 2;

. viene memorizzato nel registro H il numero 0 e nel registro L il numero che e' stato calcolato nell'accumulatore A;

. viene restituito il controllo al programma che ha mandato in esecuzione il sottoprogramma.

Riportiamo la codifica in Assembler e in codice esadecimale e decimale:

Assembler	Esadec.	Decimale	Commento
LD A,00	3E 00	62 0	Carica in A il numero che sta nel secondo byte, al= l'inizio 0.
INC A	3C	60	Incrementa A di 1.
INC A	3C	60	Incrementa A di 1.
LD H,00	26 00	38 0	Carica nel registro H il numero 0.
LD L,A	6F	111	Carica nel registro L il contenuto di A.
RET	C9	201	Restituisce il controllo al programma Basic.

Questo programma occupa 8 byte. Il risultato del calcolo va messo nella coppia di registri HL perche' cosi' vuole il Sistema Operativo dello ZX80, quando si fa uso della funzione USR per mandare in esecuzione un programma in linguaggio macchina. Questo stesso programma, per essere usato sullo ZX81 e sullo ZX80-Nuova ROM, deve essere modificato perche' in questo caso il risultato deve trovarsi nella coppia di registri BC. Si devono fare le seguenti modifiche:

```
LD H,00   diventa  LD B,00 (in codice 06 00 o 6 0)
LD L,A    diventa  LD C,A  (in codice 4F    o 79 )
```

Supponiamo di voler caricare il programma in memoria a partire dal byte 17000; i contenuti dei byte, in decimale,

per le due versioni del Basic devono essere:

Indirizzo byte	ZX80	ZX81-Nuova ROM
17000	62	62
17001	0	0
17002	60	60
17003	60	60
17004	38	6
17005	0	0
17006	111	79
17007	201	201

Dovra' essere cura del programma Basic andare a memorizzare nel byte 17001, prima di chiamare il sottoprogramma in linguaggio macchina con USR, il numero N al quale vuole aggiungere 2. Tale numero N, dovendo stare in un byte deve essere al massimo 255.

8.2. COLLEGAMENTI CON IL BASIC

Si hanno 3 possibili punti di collegamento:

.1) Istruzione: POKE n,m. Essa ci permette di scrivere nel byte di indirizzo n il valore m.

.2) Funzione: PEEK (n), senza parentesi con il nuovo Basic. Essa ci permette di leggere il contenuto del byte di indirizzo n.

.3) Funzione USR (n), senza parentesi con il nuovo Basic. Essa ci permette di andare ad eseguire una sequenza di istruzioni in linguaggio macchina, memorizzate a partire dal byte di indirizzo n. Questa funzione fornisce in una coppia di registri il risultato del calcolo se per effetto di questo il valore dei medesimi registri e' stato modificato, oppure fornisce il valore n. La coppia di registri e' HL per lo ZX80 e BC per lo ZX81 e lo ZX80-Nuova ROM.Esempio:

```
LET X=USR(17000)
```

pone X=al valore di HL o di BC oppure X=17000.

Non e' detto che il programma Basic possa usufruire solo del risultato proveniente dalla citata coppia di registri. Il programma in codice macchina puo' trasferire in zone prefissate di memoria dei dati ed il programma Basic puo' andarli a prendere usando la funzione PEEK.

Quando il programma Basic chiama la funzione USR il

sistema pone nel registro IY il numero esadecimale 4000 (corrispondente a 16384 in decimale). Questo puo' essere utile per leggere le variabili del sistema facendo uso delle istruzioni che accettano l'indirizzamento con (IY+disp).

Ricordate che sullo ZX81 funzionante in modo SLOW non si devono usare nei programmi in linguaggio macchina i registri IX e A' (registro alternativo).

8.3. COME SI CARICA IL CODICE MACCHINA

Vediamo come si puo' caricare in memoria il programma esempio del paragrafo 8.1., inserendolo in un programma Basic. Oppure...., si riferisce alla Nuova ROM.

Un primo modo, molto semplice, ma noioso se il codice macchina e' lungo, e' il seguente:

```
 10 REM PROVA ISTRUZIONI IN LINGUAGGIO MACCHINA
 20 REM SEQUENZA CARICAMENTO A PARTIRE DAL BYTE 17000
 30 REM DEL PROGRAMMA IN LINGUAGGIO MACCHINA
 40 POKE 17000,62
 50 POKE 17001,0
 60 POKE 17002,60
 70 POKE 17003,60
 80 POKE 17004,38   oppure  80 POKE 17004,6
 90 POKE 17005,0
100 POKE 17006,111  oppure 100 POKE 17006,79
110 POKE 17007,201
115 REM CHIEDE IL NUMERO INIZIALE
120 PRINT "SCRIVI UN NUMERO N <=253"
125 INPUT N
130 IF N > 253 THEN GOTO 120
135 REM STAMPA VALORE INIZIALE NUMERO
140 PRINT "VALORE INIZIALE N = ";N
145 REM SCRIVE IN 17001 IN NUMERO N
150 POKE 17001,N
155 REM VA AD ESEGUIRE ROUTINE IN CODICE MACCHINA
160 LET X = USR(17000)  oppure 160 LET X = USR 17000
165 REM STAMPA IL VALORE CALCOLATO CHE STA IN X
170 PRINT "VALORE FINALE N =";X
180 STOP
```

Il programma chiede all'utente un numero minore o uguale a 253 e lo scrive in 17001 e poi va ad eseguire la routine che aggiunge 2 ad N. Il programma in linguaggio macchina e' caricato con una serie di POKE, nelle quali e' esplicitamente scritto il numero decimale da caricare nel byte.

In questo stesso programma si potrebbe apportare la

seguente modifica:

. scrivere la linea 01 REM 0620000600600038000111201 per lo ZX80 o la linea 01 REM 0620000600600006000079201 per lo ZX81 e ZX80-Nuova ROM;

. cancellare le linee da 40 a 110;

. scrivere le seguenti linee:

```
40 LET A=16427    oppure  40 LET A=16513
45 LET M=17000
50 LET X=PEEK(A)-28
55 LET Y=PEEK(A+1)-28
60 LET Z=PEEK(A+2)-28
65 LET X=X*100+Y*10+Z
70 POKE M,X
75 IF X=201 THEN GOTO 115
80 LET A=A+3
85 LET M=M+1
90 GOTO 50
```

alla 40 si pone A al valore del primo carattere dopo la REM della linea 01; nello ZX80 i programmi iniziano a 16424 e 01REM occupa 3 byte, nell'altro sistema i programmi iniziano a 16509 e 01REM occupa 4 byte, da cui i due indirizzi citati. Nella REM della linea 01 si sono portati tutti i contenuti per i byte del programma a 3 cifre decimali aggiungendo zeri non significativi, cosi' procedendo di tre cifre per volta si hanno i valori giusti. M rappresenta l'indirizzo dove iniziare a caricare il programma in memoria. E' necessario togliere 28 ad ogni cifra prelevata dalla REM perche' i codici numerici ASCII iniziano da 28 per lo zero e poi il numero deve essere ricostruito usando le opportune potenze di 10. La sequenza di caricamento termina quando si e' caricato l'ultimo codice, che in questo caso e' 201.

Questo puo' essere un utile esempio per caricare sequenze abbastanza lunghe, qualora il programma in codice macchina sia in valori decimali. L'esempio deve essere adattato alle particolari esigenze del programma da caricare. Invece di chiedersi se l'ultimo codice caricato e' 201, si poteva istituire un contatore dicendo al programma inizialmente quanti byte dovevano essere caricati.

Si puo' usare un metodo analogo fornendo la stringa da caricare in codice esadecimale (in tale caso ogni byte viene caricato con 2 caratteri) ed usando le istruzioni seguenti, che riportiamo separatamente per i due Sistemi.

Nel programma esempio cancellare le istruzioni da 40 a

110. Per lo ZX80 procedere cosi':

. scrivere:

```
01 LET S$="3E003C3C26006FC9"
```

. scrivere:

```
35 LET M = 17000
40 LET X=CODE(S$)
45 IF X=1 THEN GOTO 115
50 LET S$ = TL$(S$)
55 LET Y = CODE(S$)
60 POKE M,16*(X-28)+Y-28
65 LET S$ = TL$(S$)
70 LET M=M+1
75 GOTO 40
```

la stringa S$ contiene il programma in esadecimale. M punta al primo byte dove caricare il programma. La 40 estrae il primo codice della stringa S$; se esso e' 1 significa che la stringa e' terminata (1=codice degli apici). In 50 la stringa S$ viene privata del suo primo carattere. In 55 viene calcolato Y, codice del secondo carattere. In 60 viene scritto un byte di programma. In 65 viene privata S$ del suo primo carattere. In 70 viene incrementato M e poi si torna al ciclo di caricamento in 40.

Usando lo stesso criterio per lo ZX81 e ZX80-Nuova ROM si deve procedere cosi:

. scrivere:

```
01 LET S$="3E003C3C06004FC9"
```

. scrivere:

```
35 LET M =17000
40 LET X=CODE S$
45 IF X=11 THEN GOTO 115
50 LET S$=S$(2 TO)
55 LET Y=CODE S$
60 POKE M,16*(X-28)+Y-28
65 LET S$=S$(2 TO)
70 LET M=M+1
75 GOTO 40
```

. oppure:

```
35 LET M=17000
40 FOR K=1 TO LEN S$ STEP 2
45 LET X=CODE (S$(K TO))
50 LET Y=CODE (S$(K+1 TO))
55 POKE M,16*(X-28)+Y-28
60 LET M=M+1
65 NEXT K
```

Si puo' caricare un programma in codice macchina usando il programma che segue, valido per lo ZX80, e, con le solite modifiche, anche per il nuovo Basic.

```
10 CLS
20 PRINT "INDIRIZZO INIZIO"
30 INPUT A
35 LET S=A
40 PRINT "PREMI NEW-LINE PER INIZIARE"
50 INPUT A$
51 CLS
55 LET I=1
60 PRINT "LOC.    HEX    DEC"
70 PRINT
80 PRINT A,
90 INPUT B$
100 PRINT B$,
105 IF B$="" THEN GOTO 300
106 IF B$="R" THEN GOTO 200
110 LET H=CODE(B$)-28
120 LET B$=TL$(B$)
130 LET L=CODE(B$)-28
140 LET T=16*H+L
150 PRINT T
160 POKE A,T
170 LET A=A+1
180 LET I=I+1
190 IF I>19 THEN GOTO 50
199 GOTO 80
200 CLS
220 PRINT "INIZIO VERIFICA ?"
230 INPUT B
235 IF B=0 THEN GOTO 300
240 CLS
241 LET I=1
242 PRINT "LOC.    HEX    DEC"
245 PRINT
246 PRINT B,
250 LET G=PEEK(B)
255 LET H=G/16+28
258 LET L=G-(H-28)*16
260 PRINT CHR$(H);CHR$(L+28),G
265 INPUT A$
270 IF A$ = "" THEN GOTO 295
280 IF A$ ="K" THEN GOTO 300
282 PRINT CHR$(137);B,
285 LET A=B
290 GOTO 90
295 LET B=B+1
296 LET I=I+1
297 IF I > 19 THEN GOTO 240
299 GOTO 246
300 PRINT "NEW-LINE PER PARTIRE"
310 INPUT A$
320 IF NOT A$="" THEN STOP
999 LET K=USR(S)
```

Il programma chiede un indirizzo iniziale da dove partire a memorizzare il codice macchina, e chiede di premere NEW-LINE per iniziare. Poi chiede il contenuto del byte in esadecimale, se si risponde solo con NEW-LINE va alla linea 300 e chiede di premere NEW-LINE per andare ad eseguire il programma; se si risponde con un altro carattere si ha uno STOP. Se al contenuto del byte si risponde con R, allora il programma prosegue dalla parte di verifica alla linea 200. Qui viene chiesto se si vuole la verifica. Se la risposta e' zero va ancora a 300, se no inizia la verifica di quanto caricato. Dopo aver listato una riga chiede un carattere, se si risponde NEW-LINE prosegue la lista, se si risponde K va a 300, se si risponde con un altro carattere si puo' correggere il contenuto dell'ultimo byte listato.

Il vantaggio di questo programma e' che consente di caricare il codice macchina in esadecimale, e che si puo' ottenere la lista anche in decimale, ma non e' molto utile per inserire tale codice in un programma, dato che esso va ricaricato digitandolo ogni volta. Puo' essere utile per fare un po' di esperienza in codice macchina aggiungendo poche frasi Basic in modo da poter effettuare delle prove.

La linea 999 puo' essere sostituita aggiungendo le frasi Basic necessarie.

8.4. ALCUNI ESEMPI IN LINGUAGGIO MACCHINA

ESEMPI PER LO ZX80

Seguono due sottoprogrammi in linguaggio macchina per ottenere sullo ZX80 lo scrolling nelle due direzioni.

Per poter provare il primo programma dovete riempire 22 linee dello schermo con 32 caratteri. Il numero di caratteri occupati nel display file per le prime 22 linee sara' 22*(32+1)=726, infatti in ogni linea dopo i 32 caratteri si ha un carattere NEW-LINE. Ricordate che le ultime 2 linee dello schermo sono a disposizione del sistema. Con questo programma perdete l'ultima linea del video e potete andare a sostituire il contenuto della prima linea che e' doppia. Segue la codifica del programma in assembler ed in codice macchina:

Assembler	Esadecimale	Decimale
LD BC, 726	01 D4 02	1 214 2
LD HL, (16396)	2A 0C 40	42 12 64
ADD, HL, BC	09	9
LD D, H	54	84
LD E, L	5D	93

```
LD BC, 693          01 B5 02        1 181  2
LD HL, (16396)      2A 0C 40       42  12 64
ADD HL, BC          09              9
LDDR                ED B8         237 184
RET                 C9            201
```

. La prima istruzione: LD BC,726 carica nel registro BC il numero dei caratteri che compongono le prime 2 linee del video.

. La seconda istruzione: LD HL, (16396) carica nel registro HL di 2 byte il contenuto del puntatore (16396 e 16397) alla memoria di schermo, quindi l'indirizzo di inizio della memoria di schermo.

. L'istruzione: ADD HL, BC calcola in HL l'indirizzo dell'ultimo byte delle 22 linee della memoria di schermo.

. Per l'istruzione LDDR occorre avere l'indirizzo che si trova in HL nei registri DE. Questo trasferimento viene fatto dalla coppia di istruzioni: LD D,L e LD E,L.

. Per procurarsi l'indirizzo dell'ultimo carattere della ventunesima riga si usano le 3 istruzioni: LD BC,693 - LD HL, (16396) - ADD HL, BC. 693=33*21 e' lo spostamento dall'inizio della memoria di tale carattere. La linea 22 va persa dato che il contenuto dello schermo si sposta verso il basso. Questo indirizzo si trova in HL.

. L'istruzione LDDR trasferisce il contenuto dell'indirizzo che sta in HL nell'indirizzo che sta in DE, poi decrementa HL e DE di 1 e decrementa anche BC di 1, fino a quando BC diventa zero. In tale modo vengono traslati i 693 caratteri in giu' sul video di una linea. La prima linea dello schermo rimane non modificata e potete andarne a modificare il contenuto usando la POKE, no la PRINT.

Per chiamare il programma, si deve memorizzare, per esempio a partire da 20000, e poi farlo eseguire scrivendo, per esempio, LET X=USR(20000). X deve essere stata definita prima nel programma Basic.

Se invece volete far muovere il contenuto dello schermo verso l'alto (scrolling normale) potete usare il programma che segue:

```
Assembler          Esadecimale      Decimale

LD BC 32            01 20 00         1 32   0
LD HL,(16396)       2A 0C 40        42 12  64
LD D,H              54              84
LD E,L              5D              93
ADD HL,BC           09               9
LD BC,693           01 BC 02         1 188  2
INC DE              13              19
LDIR                ED B0          237 176
LD (HL),118         36 76           54 118
```

RET	C9	201

il difetto e' che muovendosi in su il contenuto del video si sposta in su anche il cursore.

Nell'esempio che segue viene utilizzata la routine di stampa del sistema operativo (che inizia all'indirizzo 1376) chiamandola tramite la locazione 1824. Con questa chiamata si ottiene di andare alla routine di stampa in 1376 passandole il codice del carattere da stampare nel registro A. La stampa avviene senza errore solo se lo schermo non e' pieno. Inoltre viene sistemato il riferimento alla posizione attuale nel video. Prima di chiamare la routine tramite l'indirizzo 1824, si deve chiamare la routine di definizione della posizione attuale del cursore all'indirizzo 1760. Gli indirizzi citati sono decimali.

Assembler	Esadecimale	Decimale
LD B,128	06 80	6 128
PUSH BC	C5	197
CALL 1760	CD E0 06	205 224 6
LD A,128	3E 80	62 128
CALL 1824	CD 20 07	205 32 7
POP BC	C1	193
DJNZ,-11	10 F4	16 244
RET	C9	201

Le istruzioni seguenti servono per caricare in HL il contenuto del byte 16421, il quale contiene la posizione corrente (da 23 a 0) della linea sulla quale sta il cursore sullo schermo in fase di stampa. Segue un programma esempio, nel quale si usano queste istruzioni.

Assembler	Esadecimale	Decimale
LD HL,(16421)	2A 25 40	42 37 64
LD H,0	26 00	38 0
RET	C9	201

Caricando in HL il contenuto del byte 16421 (si pone a 0 il registro H dato che si tratta di un solo byte) si ottiene in HL il numero corrispondente alla posizione attuale della linea sullo schermo. Il byte 16421 cambia di valore solo dopo che sulla linea attuale e' stato stampato almeno un carattere. Nel programma esempio nelle linee da 1 a 6 vengono caricate le 3 istruzioni in codice macchina a partire da 30000 (si suppone di lavorare con espansione a 16K); poi da 10 a 80 vengono stampati 3 numeri su ogni linea

e questi numeri sono ottenuti con USR(30000) e quindi rappresentano il contenuto attuale del byte 16421. Dalla prova si vede che il primo dei numeri della linea si riferisce alla posizione della linea precedente. Da 100 a 150 invece si stampa un solo valore per linea e si vede che l'unico numero stampato e' il contenuto del byte 16421 riferito alla posizione della linea precedente. Segue la codifica del programma.

```
  1 POKE 30000,42
  2 POKE 30001,37
  3 POKE 30002,64
  4 POKE 30003,38
  5 POKE 30004,0
  6 POKE 30005,201
 10 PRINT
 20 FOR I=0 TO 20
 30 FOR K=1 TO 3
 40 PRINT USR(30000),
 50 NEXT K
 60 PRINT
 70 NEXT I
 80 STOP
100 PRINT
110 FOR I=0 TO 20
120 PRINT USR(30000)
130 NEXT I
140 STOP
```

Potete provare ad aggiungere la linea:

```
 25 PRINT I;" ";
```

vedrete che in questo caso i 3 numeri sulla linea sono uguali. Il programma si ferma allo STOP 80 e dovete premere 2 volte CONT e poi NEW LINE per proseguire.

ESEMPIO PER LO ZX80, PER LO ZX81 E LO ZX80-NUOVA ROM

Questi programmi servono per rinumerare da 100 con passo 10 le linee di un programma Basic, senza tener conto delle destinazioni dei GOTO/GOSUB (vedi paragrafo 9.23.).
Per la vecchia ROM:

```
        Assembler          Esadecimale       Decimale

 INIZIO LD HL,16424        21 28 40           33  40  64
        LD DE,100          11 64 00           17 100   0
 CICLO  LD BC,10           01 0A 00            1  10   0
        LD A,(HL)          7E                126
        AND 192            E6 C0             230 192
```

```
RET NZ            C0          192
LD (HL),D         72          114
INC HL            23           35
LD (HL),E         73          115
EX DE,HL          EB          235
ADD HL,BC         09            9
EX DE,HL          EB          235
LD A,118          3E 76        62 118
LD B,1            06 01         6   1
CPIR              ED B1       237 177
JR CICLO          18 EB        24 235
```

Per la nuova ROM:

```
       Assembler          Esadecimale      Decimale

INIZIO LD HL,16509        21 7D 40         33 125  64
       LD DE,100          11 64 00         17 100   0
       LD BC,10           01 0A 00          1  10   0
CICLO  LD A,(HL)          7E              126
       AND 192            E6 C0           230 192
       RET NZ             C0              192
       LD (HL),D          72              114
       INC HL             23               35
       LD (HL),E          73              115
       INC HL             23               35
       EX DE,HL           EB              235
       ADD HL,BC          09                9
       EX DE HL           EB              235
       PUSH DE            D5              213
       LD E,(HL)          5E               94
       INC HL             23               35
       LD D,(HL)          56               86
       INC HL             23               35
       ADD HL,DE          19               25
       POP DE             D1              209
       JR CICLO           18 EC            24 236
```

Potete servirvi degli indirizzi delle routine del Sistema Operativo, contenute nelle Appendici G e H per scrivere piccoli programmi che le mandino in esecuzione ed impadronirvi di molte caratteristiche del sistema.

CAPITOLO 9

E S E M P I D I
P R O G R A M M I

9.1. CONVERSIONE PROGRAMMI TRA I DIVERSI CALCOLATORI

Nei paragrafi seguenti sono riportati alcuni esempi di programmi per i calcolatori Sinclair. Dove e' significativo si riportano le modifiche da operare per poter far girare il programma sui diversi modelli. Si deve tener presente che, pur trattandosi sempre di Basic, tra le diverse implementazioni del linguaggio esistono delle differenze. Nel paragrafo 2.11. vengono elencate le differenze tra i calcolatori Sinclair e rispetto al Basic standard. Potete, con un po' di pazienza e di pratica, adattare al vostro calcolatore anche programmi scritti per altre macchine; si deve solo cercare di individuare quali sono le differenze tra le due implementazioni del linguaggio.

Passando dalla vecchia ROM alla nuova ROM e' sparita la funzione TL$; nel paragrafo 9.6 troverete un esempio di modifica di un programma.

Nella nuova ROM esistono delle operazioni in piu' rispetto allo ZX80 ed alcune istruzioni si comportano in modo diverso (RND, operatori logici), anche per questo troverete degli esempi.

Si ricorda che sullo ZX81 e ZX80-Nuova ROM non possono essere caricati nastri registrati con lo ZX80 e viceversa. Se si vogliono recuperare dei programmi, si deve ripartire dal listato.

Nella nuova ROM le variabili numeriche occupano piu' spazio; per questo un programma che sta in 1K con la vecchia ROM puo' non entrare in 1K con la nuova ROM. Nella nuova ROM l'utilizzo della memoria e' diverso: se in alcuni programmi si contano i byte a partire dall'inizio di una REM o di una PRINT si devono rifare i conti.

Nella nuova ROM per contare intervalli di tempo si ha l'istruzione PAUSE; PAUSE 50 tiene fermo lo schermo per 1 secondo, PAUSE 25 per mezzo secondo.

Nella nuova ROM si ha liberta' di movimento sullo schermo.

Le divisioni nei programmi vecchia ROM danno risultati interi; nella nuova ROM per ottenere lo stesso risultato si deve usare la funzione INT.

9.2. DIVISIONE CON DECIMALI SULLO ZX80

Lo ZX80 lavora solo con numeri interi, con questo programma si ottiene il risultato di una divisione con 3 decimali. Ovviamente non ha senso trasformare il programma per la nuova ROM. (1K).

Analisi del problema:

.a) Vengono richiesti il dividendo e il divisore.
.b) La variabile X contiene il dividendo e la variabile Y il divisore.
.c) Viene calcolata la parte intera Z del quoziente.
.d) Viene calcolato il resto R1. Il primo decimale D1 e' ottenuto moltiplicando il resto R1 per 10 e poi dividendolo per Y.
.e) Viene calcolato il nuovo resto R2. Il secondo decimale D2 e' ottenuto moltiplicando il resto R2 per 10 e dividendo poi per 10.
.f) Viene calcolato il nuovo resto R3. Il terzo decimale D3 viene ottenuto moltiplicando il resto R3 per 10 e poi dividendo per Y.
.g) Si stampa il risultato Z. D1 D2 D3.

Codifica del programma:

```
 10 REM DIVISIONE CON TRE DECIMALI
 15 PRINT "DIVISIONE CON TRE DECIMALI"
 20 PRINT "DIVIDENDO = ?"
 30 INPUT X
 40 PRINT "DIVISORE = ?"
 50 INPUT Y
 60 LET Z=X/Y
 70 LET R1 = X - Z * Y
 80 LET D1 = 10 * R1/Y
 90 LET R2 = 10 * R1 - D1 * Y
100 LET D2 = 10 * R2/Y
110 LET R3 = 10 * R2 - D2 * Y
120 LET D3 = 10 * R3/Y
130 PRINT "RISULTATO:";Z;".";D1;D2;D3
```

Il programma che segue calcola invece la divisione tra due interi con il numero N di decimali desiderato.

Analisi del problema:

.a) Si richiede il numero di decimali desiderato e si memorizza in D.
.b) Si richiede il dividendo e si memorizza in R.
.c) Si pone K = R, cioe' K contiene il dividendo.
.d) Si richiede il divisore e si memorizza in Y.

.e) Si calcola Z = R/Y, Z e' la parte intera del quoziente; si calcola R = R - Z * Y, cioe' si sostituisce al dividendo iniziale il primo resto trovato.

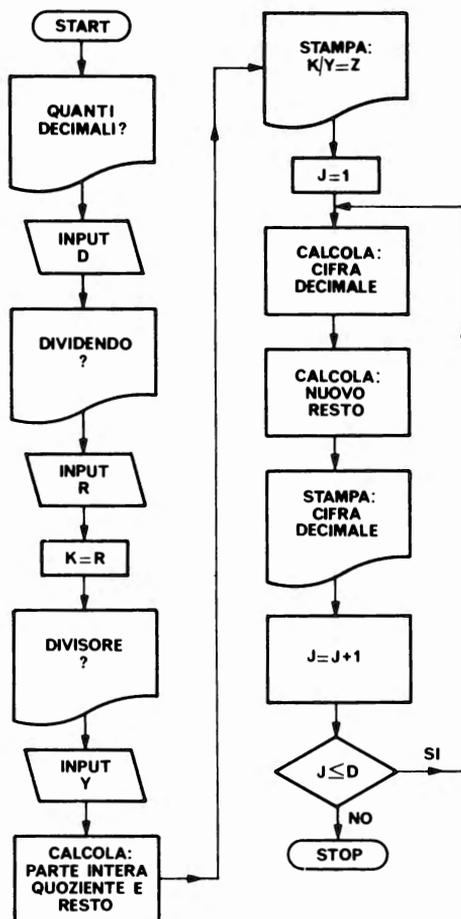
.f) Si scrive la prima parte del risultato senza andare a capo.

.g) Si inizia il ciclo di calcolo per i D decimali.

.h) Si calcola Z = 10 * R/Y cioe' si moltiplica il resto per 10 e poi si divide per il divisore; si calcola il nuovo resto e si sostituisce in R al vecchio.

.i) Si stampa Z, cifra decimale calcolata.

.l) Se il ciclo non e' finito si torna al punto h) dopo aver incrementato la variabile J che controlla il ciclo.

Diagramma a blocchi:

Codifica del programma:

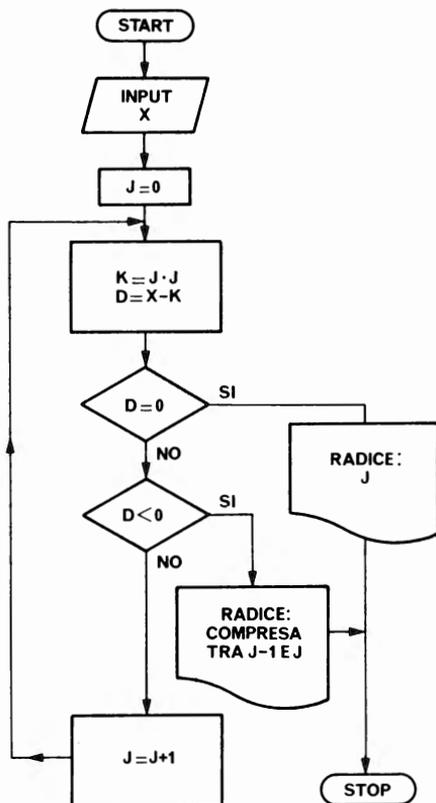
```
 10 REM DIVISIONE AD ALTA PRECISIONE
 15 PRINT "DIVISIONE AD ALTA PRECISIONE"
 20 PRINT "QUANTI DECIMALI ?"
 30 INPUT D
 40 PRINT "DIVIDENDO ?"
 50 INPUT R
 55 LET K = R
 60 PRINT "DIVISORE ?"
 70 INPUT Y
 80 LET Z = R/Y
 90 LET R = R - Z * Y
 95 PRINT
100 PRINT K;"/";Y;"=";",";
110 FOR J = 1 TO D
120 LET Z = 10 * R/Y
130 LET R = 10 * R - Z * Y
140 PRINT Z;
150 NEXT J
```

9.3. CALCOLO RADICE QUADRATA

Questo programma serve per calcolare la radice quadrata di un numero sullo ZX80; il risultato viene dato segnalando i due interi tra i quali e' compresa la radice cercata. (1K).

Analisi del problema:

.a) Viene richiesto il numero e memorizzato in X.
.b) Viene inizializzata al valore zero la variabile J, tale variabile viene poi incrementata di 1 ad ogni ciclo per trovare la radice di X.
.c) Inizia il calcolo ciclico: si calcola K = J * J.
.d) Si calcola D = X - K.
.e) Se D = 0 si stampa J, radice di X e si va allo STOP.
.f) Se D risulta minore di zero allora non esiste una radice intera esatta, si stampa che la radice e' compresa tra J - 1 e J e si va allo STOP.
.g) Se D non risulta minore di zero, si incrementa J di 1 e si torna al punto c).
.h) Si ferma il programma.

Diagramma a blocchi:

Codifica del programma:

```
 10 REM CALCOLO RADICE QUADRATA
 15 PRINT "CALCOLO RADICE QUADRATA"
 20 PRINT "SCRIVI IL NUMERO"
 30 INPUT X
 40 LET J = 0
 50 LET K = J * J
 60 LET D = X - K
 70 IF D = 0 THEN GO TO 110
 80 IF D < 0 THEN GO TO 130
 90 LET J = J + 1
100 GO TO 50
110 PRINT "RADICE ";J
120 GO TO 140
```

```
130 PRINT "RADICE COMPRESA TRA ";(J-1);" E ";J
140 STOP
```

Non ha senso trasformare il programma per la nuova ROM, dal momento che si puo' estrarre la radice quadrata da qualunque numero con il seguente semplice programma:

```
10 REM CALCOLO RADICE QUADRATA X
20 PRINT "SCRIVI UN NUMERO X"
30 INPUT X
40 PRINT "LA RADICE QUADRATA DI ";X;" E' ";SQRX
50 STOP
```

9.4. LANCIO DEI DADI

Questo programma simula il lancio di un dado sfruttando la funzione RND.

Versione valida sullo ZX80.

Analisi del problema:

.a) Si preparano delle stringhe contenenti i caratteri grafici necessari per poter evidenziare i dadi sullo schermo.
.b) Si inizia la sequenza di ricerca di un numero pseudo-random <= 6.
.c) A seconda del numero si salta al pezzo di programma che disegna il dado uscito e poi si torna sempre al punto d).
.d) Si chiede di premere NEW LINE per lanciare ancora, si analizza il tasto premuto e se e' NEW LINE si torna al punto b) dopo aver azzerato lo schermo, se non si vuole piu' lanciare si preme un qualunque altro tasto ed il programma si ferma.

Codifica del programma:

```
 15 PRINT "LANCIO DEI DADI"
 20 LET A$ = ".   ."
 30 LET B$ = "  .  "
 40 LET C$ = "    ."
 50 LET D$ = ".    "
 60 LET E$ = "     "
120 LET X = RND (6)
135 PRINT
136 PRINT
140 IF X = 1 THEN GO TO 200
```

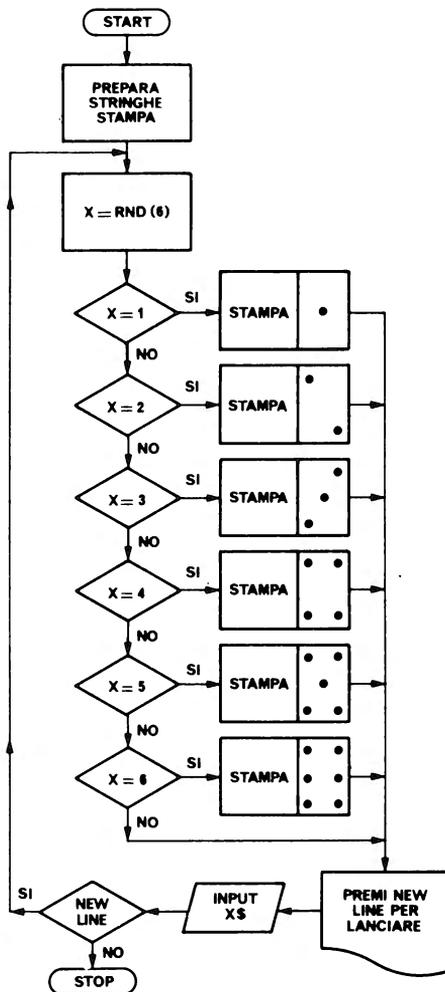
```
 150 IF X = 2 THEN GO TO 300
 160 IF X = 3 THEN GO TO 400
 170 IF X = 4 THEN GO TO 500
 180 IF X = 5 THEN GO TO 600
 190 IF X = 6 THEN GO TO 700
 195 GO TO 1000
 200 PRINT E$
 205 PRINT E$
 210 PRINT B$
 215 PRINT E$
 220 PRINT E$
 230 GOTO 1000
 300 PRINT C$
 305 PRINT E$
 310 PRINT E$
 315 PRINT E$
 320 PRINT D$
 330 GOTO 1000
 400 PRINT D$
 405 PRINT E$
 410 PRINT B$
 415 PRINT E$
 420 PRINT C$
 430 GOTO 1000
 500 PRINT A$
 505 PRINT E$
 510 PRINT E$
 515 PRINT E$
 520 PRINT A$
 530 GOTO 1000
 600 PRINT A$
 605 PRINT E$
 610 PRINT B$
 615 PRINT E$
 620 PRINT A$
 630 GOTO 1000
 700 PRINT A$
 705 PRINT E$
 710 PRINT A$
 715 PRINT E$
 720 PRINT A$
1000 PRINT
1001 PRINT
1002 PRINT
1003 PRINT
1010 PRINT "PREMI (NEW LINE) PER LANCIARE"
1011 PRINT "ANCORA"
1100 INPUT X$
1200 CLS
1300 IF X$ = "" THEN GO TO 120
```

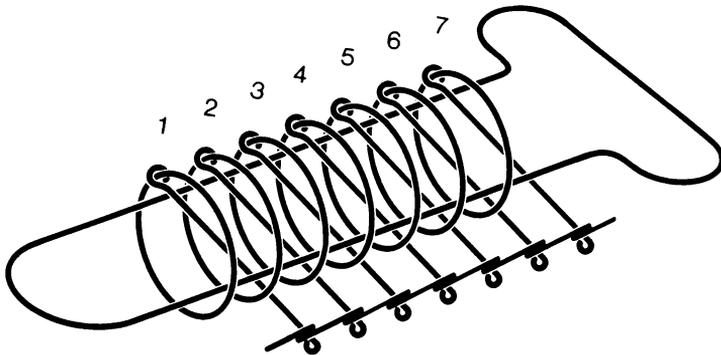
Per far funzionare il programma sullo ZX81 o ZX80-Nuova

ROM si deve solo modificare l'istruzione 120:

```
120 LET X =INT(1 + 6 * RND)
```

infatti il numero a caso e' minore di 1 e quindi va moltiplicato per 6 per ottenere il valore della faccia del dado.

Diagramma a blocchi:

9.5. GIOCO DEGLI ANELLI CINESI

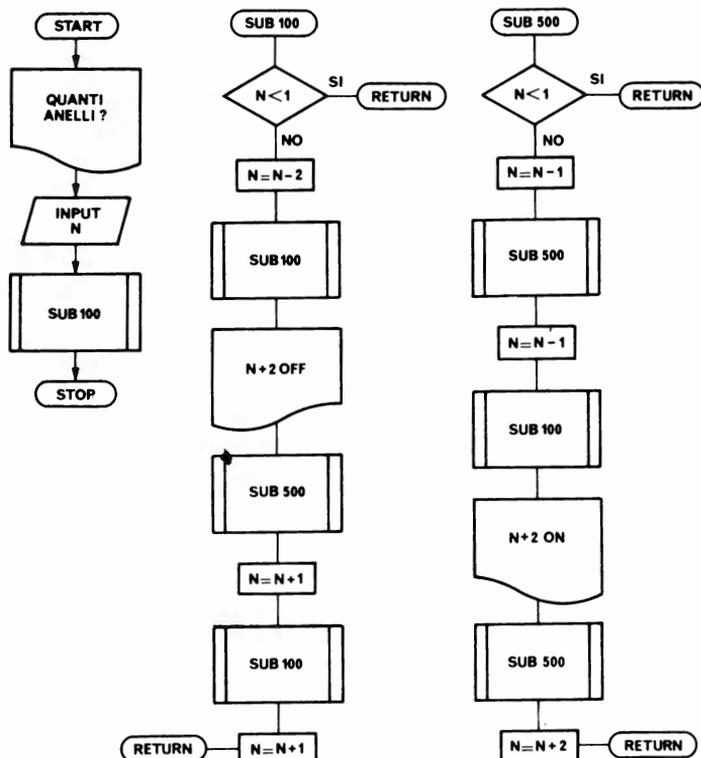
Questo programma gira sui 3 calcolatori. (1K). Esso simula il gioco degli Anelli Cinesi, vedi figura che segue.

Il gioco consiste nel riuscire a togliere il numero stabilito di anelli, rispettando le seguenti regole:

.a) Si puo' muovere un anello per volta.
.b) Il primo anello puo' essere tolto in qualsiasi momento.
.c) L'anello di posto I (I>1) puo' essere tolto o messo se e solo se:

- tutti gli anelli fino al posto I-2 sono stati tolti;
- l'anello di posto I-1 e' al suo posto;
- gli anelli di posto >I possono essere in qualunque stato.

Si dice che un anello e' ON quando e' montato, che e' OFF quando e' smontato.

Il programma si articola in un gioco di chiamate a due sottoprogrammi interni che alternativamente si richiamano o richiamano se stessi. Si ha come output l'elenco delle mosse da fare. Per togliere il settimo anello, le mosse sono molte e non sono contenute tutte nello schermo, e' necessario ricorrere al tasto CONT per vederle tutte.

Diagramma a blocchi:

Codifica del programma:

```
 10 REM ANELLI CINESI
 15 PRINT "ANELLI CINESI"
 20 PRINT "QUANTI ANELLI VUOI TOGLIERE"
 25 INPUT N
 30 GO SUB 100
 40 STOP
100 IF N < 1 THEN RETURN
120 LET N = N - 2
130 GO SUB 100
140 PRINT N + 2;"OFF",
150 GO SUB 500
160 LET N = N + 1
170 GO SUB 100
180 LET N = N + 1
190 RETURN
500 IF N < 1 THEN RETURN
```

```
520 LET N = N - 1
530 GO SUB 500
540 LET N = N - 1
550 GO SUB 100
560 PRINT N + 2;"ON",
570 GO SUB 500
575 LET N = N + 2
580 RETURN
```

9.6. CARATTERI IN CAMPO INVERSO

Di questo programma si riporta la codifica per lo ZX80 e quella per lo ZX81 e ZX80-Nuova ROM. (1K).

Analisi del problema:

.a) Viene richiesta una stringa alfanumerica e memorizzata in G$.
.b) Viene stampata la stringa letta.
.c) Inizia il ciclo di trasformazione dei caratteri componenti la stringa, tale ciclo termina quando si incontra la fine della stringa (stringa nulla, CHR $(1));ogni carattere viene decodificato, il codice viene modificato aggiungendo 128 e cosi' diventa il carattere in campo inverso, poi viene riconvertito in stringa e stampato.

Codifica del programma per lo ZX80:

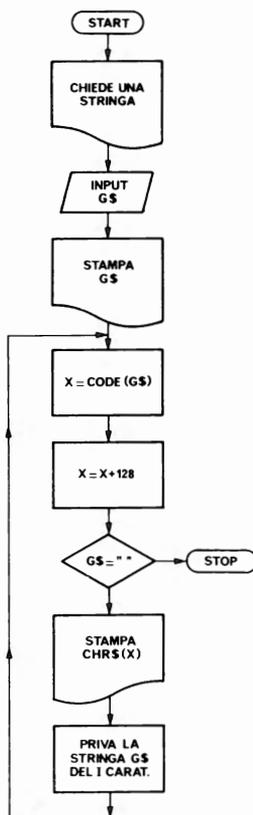
```
 10 REM PROVA CARATTERI
 11 REM IN CAMPO INVERSO
 15 PRINT "SCRIVI UN CARATTERE"
 20 PRINT "O ALCUNI CARATTERI"
 25 INPUT G$
 30 PRINT "HAI SCRITTO:";G$
 35 PRINT "RISCRIVO IN CAMPO INVERSO"
 40 LET X = CODE(G$)
 50 LET X = X + 128
 60 IF G$ = CHR$(1) THEN GO TO 100
 70 PRINT CHR$(X);
 80 LET G$ = TL$(G$)
 90 GO TO 40
100 STOP
```

Codifica del programma per lo ZX81 e lo ZX80-Nuova ROM:

```
 10 REM PROVA CARATTERI
 11 REM IN CAMPO INVERSO
 15 PRINT "SCRIVI UN CARATTERE"
 20 PRINT "O ALCUNI CARATTERI"
 25 INPUT G$
 30 PRINT "HAI SCRITTO: ";G$
```

```
 35 PRINT "RISCRIVO IN CAMPO INVERSO"
 38 FOR K=1 TO LEN(G$)
 40 LET X=CODEG$
 50 LET X=X+128
 60 IF G$ = CHR$1 THEN GO TO 100
 70 PRINT CHR$X;
 80 LET G$ = G$(2 TO)
 90 NEXT K
100 STOP
```

Confrontando le due codifiche potete vedere come si possa fare a meno della TL$ sfruttando le istruzioni di "slicing" e la funzione LEN.

Diagramma a blocchi:

9.7. GRAFICO DI DUE FUNZIONI SULLO ZX80

Questo programma e' stato scritto per lo ZX80 (1K), con pochi cambiamenti nei codici dei caratteri grafici gira anche sullo ZX81 e ZX80-Nuova ROM. Con la ROM da 8K si puo' programmare piu' agevolmente il grafico di una funzione servendosi dei nuovi comandi disponibili.

Analisi del problema:

Le due funzioni sono Y = X e Z = 24 - X; i valori di Z vengono stampati in nero e quelli Y in grigio. A seconda del valore della variabile J, che controlla il ciclo piu' interno, e dei valori Y e Z viene scelto il carattere da stampare scegliendo tra i seguenti:

- per la vecchia ROM: CHR$(3), CHR$(11), CHR$(139);
- per la nuova ROM: CHR$(131), CHR$(10), CHR$(138).

Codifica per la vecchia ROM:

```
  5 REM GRAFICI DI DUE FUNZIONI
 10 LET X = 0
 20 PRINT "GRAFICI DI DUE FUNZIONI"
 30 PRINT "X ="
 40 FOR I = 1 TO 21
 50 LET Y = X
 60 LET Z = 24 - X
 70 PRINT X
 80 FOR J = 1 TO 20
 85 IF J > Y AND J = Z THEN PRINT CHR$(3);
 90 IF J > Y AND J > Z THEN GO TO 135
 95 IF J = Y AND J > Z THEN PRINT CHR$(11);
100 IF J < Y AND J < Z THEN PRINT CHR$(139);
110 IF J < Y AND J > Z THEN PRINT CHR$(11);
115 IF J = Y AND J < Z THEN PRINT CHR$(139);
120 IF J > Y AND J < Z THEN PRINT CHR$(3);
125 IF J = Y AND J = Z THEN PRINT CHR$(139);
130 NEXT J
135 PRINT
140 LET X = X + 1
150 NEXT I
```

Codifica per la nuova ROM:

```
Basta modificare: CHR$(3)   diventa CHR$(131);
                  CHR$(11)     "    CHR$(10);
                  CHR$(139)    "    CHR$(138).
```

9.8. TABULAZIONE E GRAFICO FUNZIONE SULLO ZX81 E SULLO ZX80-NUOVA ROM

Questo programma tabula una funzione e ne traccia il grafico nell'intervallo X1-X2 che gli viene fornito. L'incremento usato e' uguale a 1. L'utente deve modificare la linea 110 del programma inserendo la formula che calcola la funzione di X. L'utente deve scrivere la formula in modo tale che usando l'incremento di una unita' abbia senso tracciare il grafico tra i due valori limite assegnati. Si ricordi che l'istruzione PLOT lavora su valori interi. (1K).

Codifica del programma:

```
 10 PRINT "TABULAZIONE E GRAFICO FUNZIONE"
 20 PRINT "VALORE MINIMO X?"
 30 INPUT X1
 40 PRINT "VALORE MASSIMO X?"
 50 INPUT X2
 60 CLS
 70 LET Z=X2-X1
 80 DIM Y(Z)
 90 FOR X=X1 TO X2
100 SCROLL
110 LET Y(X) = ..............................
120 PRINT X,Y(X)
130 NEXT X
140 PAUSE 300
150 CLS
160 FOR X=X1 TO X2
170 PLOT X,Y(X)
180 NEXT X
```

Note al programma:

. La linea 110 va riscritta prima di dare il RUN del programma; si puo' scrivere per esempio:

```
         110 LET Y(X) = X*X/100
oppure   110 LET Y(X) = 6*SQR(X)
oppure   110 LET Y(X) = 19*COS X
oppure   110 LET Y(X) = 3.5*X**2
```

naturalmente alla richiesta del minimo e del massimo per X si deve dare una risposta che abbia senso relativamente alla funzione da calcolare. Negli esempi di cui sopra potete provare tra 1 e 60.

. Nelle linee da 10 a 50 vengono chiesti e memorizzati i valori limite X1 e X2.

. La linea 60 pulisce lo schermo.

. La linea 70 calcola le dimensioni dei dati da

memorizzare.

. La linea 80 dimensiona il vettore per memorizzare i valori di Y.

. Dalla linea 90 alla linea 130 viene tabulata la funzione sul video e vengono memorizzati i valori. Dato che alla linea 100 si usa la istruzione SCROLL, se i risultati sono tanti si perdono i primi per effetto del movimento dello schermo.

. Alla linea 140 si ha una PAUSE per consentire di leggere i risultati; si puo' aumentare il tempo della pausa modificando il 300.

. Alla linea 150 viene pulito lo schermo.

. Dalla linea 160 alla linea 180 si ha il ciclo per tracciare il grafico.

Sul calcolatore ZX81 il programma puo' essere provato sia in modo FAST che in modo SLOW.

9.9. CALCOLO MEDIA, VARIANZA E DEVIAZIONE STANDARD SULLO ZX81 E ZX80-NUOVA ROM

Questo programma consente di calcolare la media, la varianza e la deviazione standard di N dati, con N<=50. L'utente deve fornire in INPUT i dati e la loro frequenza, nell'ordine: dato, frequenza. Il programma enumera sul video i dati forniti. Se l'utente si accorge di aver commesso un errore di dato puo' rispondere alla richiesta di dato con E (errore) ed il programma chiede nuovamente il dato e la relativa frequenza. Per chiudere l'immissione di dati si deve rispondere con la lettera T (tappo). Se l'utente desidera immettere piu' di 50 dati, deve solo modificare la linea 5 e porre la variabile N che serve per dimensionare i due vettori X (per i dati) e F (per le frequenze) al numero di elementi desiderato. Le formule di calcolo usate sono le seguenti:

Media = (Sommatoria X)/(Sommatoria F)

Varianza = (Sommatoria X**2)/(Sommatoria F) - Media**2

Deviazione standard = Radice quadrata (Varianza)

Codifica programma:

```
 5 LET N = 50
 6 DIM X(N)
 7 DIM F(N)
10 PRINT "ARG.----DATO------FREQUENZA",
11 PRINT "==========================="
```

```
 20 LET E = RND
 30 LET T = RND
 40 LET A = 0
 50 LET B = 0
 60 LET A = A + 1
 70 LET B = B + 1
 80 IF B > 19 THEN SCROLL
 90 PRINT A;
100 INPUT X
110 IF X = E THEN GOTO 200
120 IF X = T THEN GOTO 260
130 LET X(A) = X
140 PRINT TAB 10;X;
150 INPUT F(A)
160 PRINT TAB 21;F(A)
170 GOTO 60
200 LET B = B + 1
210 IF B > 19 THEN SCROLL
220 PRINT TAB 0;"ARG.?"
230 INPUT A
250 GO TO 70
260 LET Z = 0
270 LET A = 0
280 LET B = 0
290 FOR K = 1 TO N
300 LET Z = Z + F(K)
310 LET A = A + F(K)*X(K)
320 LET B = B + F(K)*X(K)*X(K)
330 NEXT K
340 CLS
350 PRINT AT 5,0;"MEDIA=",,,A/Z,,,"VARIANZA=",,,
    B/Z - A*A/(Z*Z),,,"DEVIAZIONE STANDARD-",,
    SQR(B/Z - A*A/(N*N))
```

Elenco variabili usate nel programma:

N numero massimo elementi trattato;
X(N) vettore per i dati;
F(X) vettore per le frequenze;
E variabile come nome riferimento per errore;
T variabile come nome riferimento per tappo;
A contatore dati;
B contatore linee video;
X dato in ingresso;
Z sommatoria frequenze e quindi conteggio numero totale dati;
K variabile contatore del ciclo.

Note al programma:

. Linee 5 - 7 inizializzazione numero letture e dimensionamento vettori.

. Linea 10 stampa testata.

. Linee 20 - 30 creazione variabili di riferimento per errore e tappo; da notare l'uso particolare che si fa di queste variabili negli IF delle linee 110 e 120.

. Linee 40 - 50 inizializzazione contatori A e B.

. Linea 60 inizio ciclo per A.

. Linea 70 inizio ciclo per B.

. Linea 80 eventuale SCROLL del video.

. Linea 90 stampa contatore letture

. Linee 100-130 lettura dato con controllo errore o tappo e memorizzazione.

. Linee 140 - 160 lettura frequenza e completamento linea video.

. Linea 170 ritorna al ciclo di A.

. Linee 180 - 250 se errore chiede un nuovo dato; attenzione, si puo' correggere l'ultimo dato introdotto. Se si vuol correggere un dato precedente si puo', ma poi si deve ridare ancora errore e ridare l'argomento da cui proseguire.

. Linee 260 - 280 inizializzazione variabili per calcoli.

. Linee 290 - 330 ciclo di calcolo.

. Linea 340 pulizia video.

. Linea 350 stampa risultati, notare che le espressioni vengono calcolate in fase di stampa.

Si fa notare che l'uso delle variabili E e T non e' proprio del Basic standard. Il sistema accetta come INPUT numerico nella variabile X (numerica) la risposta sotto forma di E o di T, mentre darebbe errore per un'altra lettera non gia' definita come variabile nel programma. Se voi andate ad analizzare il contenuto di X, dopo la risposta E, lo troverete identico al contenuto di E. Questo significa che il sistema, ricevuta la risposta E (o T chiaramente) va a a scandire le variabili del programma e quando incontra E pone il suo contenuto in X. Da cui si ricava che le linee 20 e 30 avrebbero potuto essere anche diverse, come E = 1 e T = 0 o altro, basta che le variabili E e T siano inizializzate in qualche modo, cioe' incomincino ad esistere. Se non siete convinti potete provare questo semplicissimo programmino:

```
10 LET A = 1
20 INPUT X
30 IF X = A THEN PRINT "UGUALE AD A ";A,X
40 PRINT "ESEGUITO"
```

se rispondete con un qualunque numero alla richiesta di INPUT vedrete sul video ESEGUITO; se invece rispondete con A, vedrete:

```
UGUALE AD A  valore di A      valore di X
```

e verificherete che i due valori sono uguali.

Si fa notare che anche lo ZX80 con la vecchia ROM ha lo stesso comportamento.

9.10. RISOLUZIONE EQUAZIONE IN X SULLO ZX81 E ZX80-NUOVA ROM

Con questo programma si puo' risolvere una equazione in X fornendola al programma come stringa in fase di utilizzo. Il programma richiede due numeri che l'utente pensa si avvicinino alla soluzione e calcola l'errore fatto nella previsione. Viene sfruttata la capacita' della funzione VAL di operare su una stringa che rappresenta una espressione aritmetica. Nel rispondere con la stringa, questa va scritta usando gli stessi tasti che si userebbero per scrivere una linea di programma; cioe' una funzione non va scritta lettera per lettera, ma va usato il tasto apposito.

Codifica del programma:

```
 10 PRINT "SCRIVERE UNA FUNZIONE DI X"
 20 PRINT "0 = ";
 30 INPUT F$
 40 PRINT F$
 50 PRINT
 60 PRINT "SCRIVETE DUE NUMERI ENTRO I QUALI PENSATE
           SIA COMPRESO IL RISULTATO"
 70 INPUT X1
 80 PRINT
 90 PRINT X1,
 95 LET X = X1
 97 LET F = VAL F$
100 IF F <> 0 THEN GOTO 130
110 PRINT AT 13,3;"SOLUZIONE",X1
120 STOP
130 INPUT X2
140 PRINT X2
150 PRINT AT 10,10;"RIS.="
160 PRINT AT 11,10;"ERR.="
170 LET X = X2
180 LET G = VAL F$
190 PRINT AT 10,17;X1
200 PRINT AT 11,17;ABS (G-F)
210 IF ABS (F-G)>1E-9 AND G<>0 THEN GOTO 240
220 LET X1 = X2
230 GOTO 110
240 LET X = (G*X1 - F*X2)/(G-F)
250 LET F = G
260 LET X1 = X2
```

```
270 LET X2 = X
280 GOTO 180
```

Elenco variabili usate nel programma:

. F$ stringa contenente l'equazione in X da risolvere.
. X1 variabile contenente l'estremo sinistro dell'intervallo per X.
. X2 variabile contenente l'estremo destro dell'intervallo per X.
. F variabile per calcolo funzione con X = X1 iniziale o calcolato.
. G variabile per calcolo funzione con X = X2 iniziale o calcolato.
. X variabile usata per il calcolo della funzione.

Note al programma:

. Nelle linee da 10 a 50 viene richiesta la equazione da calcolare.
. Nella linea 60 vengono richiesti i due valori limiti X1 e X2.
. Nelle linee da 70 a 90 viene letto e stampato X1.
. Nelle linee da 95 a 120 viene calcolata la funzione F per X = X1 e se la funzione risulta = 0 viene stampato il risultato finale per X = X1 ed il programma si ferma. Se invece F risulta diversa da 0 il programma va a richiedere X2.
. Nelle linee da 130 a 140 viene letto e stampato X2.
. Le linee 150 e 160 preparano la stampa del risultato in base ai valori proposti e all'errore commesso.
. Nelle linee 170 e 180 viene calcolata la funzione G per X = X2.
. Nelle linee 190 e 200 vengono stampati i valori RIS. e ERR..
. La linea 210 controlla se il risultato e' accettabile: se si viene posto X1 = X2 e proposto tale risultato come SOLUZIONE ed il programma si ferma.
. Se il risultato non e' accettabile viene calcolato un valore approssimato per X; viene posto F = G, X1 = X2 e X2 = X e ricomincia il calcolo di G.

Si fa notare che l'uso della funzione VAL presentato in questo programma non e' standard nelle implementazioni del Basic ed arricchisce notevolmente le possibilita' del linguaggio. Se non siete completamente convinti, provate a scrivere questo semplicissimo programmino e provatelo.

```
10 INPUT X          chiede un valore per X;
20 INPUT F$         chiede la formula da calcolare;
30 PRINT F$         stampa la formula introdotta;
```

```
40 PRINT VAL F$        stampa il valore calcolato;
```

dopo il RUN del programma scrivete in modo immediato:

```
PRINT seguito dalla formula gia' introdotta;
```

confrontate con il risultato ottenuto precedentemente. Naturalmente la formula deve essere scritta usando la variabile X e seguendo la sintassi del Basic.

9.11. PRONTEZZA DEI RIFLESSI

Di questo programma si riportano le due versioni valide sulle due ROM. (1K).

Analisi del problema:

.a) All'inizio viene creato un ciclo di attesa per rendere possibile l'operazione.
.b) Vengono azzerati i 2 bytes che costituiscono il contatore dei fotogrammi del video.
.c) Viene richiesto di schiacciare NEW LINE.
.d) Viene memorizzata la risposta in C$.
.e) Viene memorizzato il valore dei due byte del contatore.
.f) Viene calcolato il valore del contatore; viene tolto 4 perche' si presuppone un ritardo di 80 millisecondi nell'arrivo della risposta. Viene stampato il tempo di risposta.

Codifica per lo ZX80 vecchia ROM:

```
 5 REM TEMPO DI RISPOSTA
10 PRINT "TEMPO DI RISPOSTA"
15 FOR I = 1 TO 20 * RND(100)
20 NEXT I
30 POKE 16414,0
40 POKE 16415,0
50 PRINT "SCHIACCIA (NEW LINE)"
60 INPUT C$
70 LET A = PEEK(16414)
80 LET B = PEEK(16415)
90 PRINT "TUO TEMPO DI RISPOSTA: ";(B*256+A-4)*20;
   " MILLISECONDI"
```

Nella vecchia ROM l'indirizzo del contatore dei fotogrammi dello schermo e' 16414 (e quindi 16414/16415).

Codifica per lo ZX81 (modo FAST) e ZX80-Nuova ROM:

```
 5 REM TEMPO DI RISPOSTA
10 PRINT "TEMPO DI RISPOSTA"
15 FOR I = 1 TO 200 * RND
20 NEXT I
30 POKE 16436,0
40 POKE 16437,0
50 PRINT "SCHIACCIA (BREAK)"
70 LET A = PEEK 16436
80 LET B = PEEK 16437
90 PRINT "TUO TEMPO DI RISPOSTA: ";(B*256+A-4)*20;
          " MILLISECONDI"
```

Nella nuova ROM l'indirizzo del contatore dei fotogrammi dello schermo e' 16436 (e quindi 16436/16437). Il contatore dei fotogrammi dello schermo viene modificato anche dalla istruzione PAUSE. Il contatore dei fotogrammi dello schermo viene modificato se si invia un messaggio al video.

9.12. MORSI NEL FORMAGGIO

Questo programma puo' girare su ambedue i calcolatori, pur di modificare le linee 500 e 510 per lo ZX81 e ZX80-Nuova ROM.

Analisi del problema:

.a) All'inizio vengono riempiti di 1 A(10), B(10) e C(10); questi 3 vettori rappresentano le 3 righe che vengono via via disegnate sul video. In seguito viene analizzato ogni vettore e quindi ogni riga e, se l'elemento e' 1, viene stampato un quadratino nero (CHR$(128)), mentre se l 'elemento e' zero si ha uno spazio.
.b) Si analizza il vettore A e si stampa la prima riga.
.c) Si analizza il vettore B e si stampa la seconda riga.
.d) Si analizza il vettore C e si stampa la terza riga.
.e) Viene chiesto di premere NEW LINE per mordere il formaggio. Se si preme un altro tasto il programma va allo STOP. Se si preme NEW LINE si generano due numeri pseudo-random I <= 10 e K <= 3; tali numeri sono le coordinate dell 'elemento da azzerare in uno dei tre vettori si torna quindi al ciclo di stampa b).

Codifica per lo ZX80:

```
 5 REM MORSI NEL FORMAGGIO
10 PRINT "MORSI NEL FORMAGGIO"
12 PRINT
```

```
  15 DIM A(10)
  20 DIM B(10)
  30 DIM C(10)
 100 FOR J = 1 TO 10
 110 LET A(J) = 1
 120 LET B(J) = 1
 130 LET C(J) = 1
 140 NEXT J
 200 FOR J = 1 TO 10
 205 IF NOT A(J) = 1 THEN GOTO 220
 210 PRINT CHR$(128);
 215 GOTO 230
 220 PRINT " ";
 230 NEXT J
 240 PRINT
 300 FOR J = 1 TO 10
 305 IF NOT B(J) = 1 THEN GOTO 320
 310 PRINT CHR$(128);
 315 GOTO 330
 320 PRINT " ";
 330 NEXT J
 340 PRINT
 400 FOR J = 1 TO 10
 405 IF NOT C(J) = 1 THEN GOTO 420
 410 PRINT CHR$(128);
 415 GOTO 430
 420 PRINT " ";
 430 NEXT J
 440 PRINT
 442 PRINT"PREMI (NEW LINE) PER MORDERE
       IL FORMAGGIO"
 450 INPUT Y$
 460 IF NOT Y$ = "" THEN GO TO 100
 470 CLS
 500 LET I = RND(10)
 510 LET K = RND(3)
 520 IF K = 1 THEN LET A(I) = 0
 530 IF K = 2 THEN LET B(I) = 0
 540 IF K = 3 THEN LET C(I) = 0
 550 GOTO 200
1000 STOP
```

Codifica per lo ZX81 e lo ZX80-Nuova ROM:

Modificare le due linee 500 e 510 cosi':

```
 500 LET I = INT(RND*10) + 1
 510 LET K = INT(RND*2) + 1
```

Provate a far girare il programma sullo ZX81 nei due modi FAST e SLOW.

9.13. INGRANDIMENTO CARATTERI

Per ingrandire i caratteri si possono usare le matrici 8x8 dei caratteri memorizzate in ROM. Per la vecchia ROM le matrici dei caratteri iniziano al byte 3584, mentre per la nuova ROM iniziano al byte 7680. Dando un passo di 8 si hanno gli 8 byte che servono al sistema per visualizzare i caratteri sullo schermo. Tali matrici 8x8 possono essere usate come maschere di stampa per ottenere i caratteri ingranditi, facendo corrispondere, in base ad un modulo prefissato, ai bit 0 un certo numero di spazi ed ai bit 1 un carattere grafico ripetuto un certo numero di volte.
Si riportano due programmi di ingrandimento, uno per la vecchia ROM ed uno per la nuova ROM.

Codifica di un programma per lo ZX80:

```
 10 CLS
 15 PRINT "   ***CARATTERI 8X8***"
 18 PRINT
 20 DIM P(7)
 25 PRINT "PER LA TAB. CODICI DARE UN COD.<0"
 27 PRINT
 30 PRINT"COD. CARATTERE >=0"
 40 INPUT X
 45 IF X<0 THEN GO TO 500
 50 FOR I = 0 TO 7
 60 LET P(I) = 2**(7-I)
 70 NEXT I
 90 FOR I = 0 TO 7
100 LET V = PEEK (3584 + I + 8*X)
110 FOR K = 0 TO 7
120 LET G = (V AND P(K))>0
130 PRINT CHR$(-128*G);
140 NEXT K
150 PRINT
160 NEXT I
170 PRINT "ANCORA S/N?"
180 INPUT R$
190 IF R$ = "N" THEN STOP
200 GO TO 10
500 LET R=1
505 CLS
510 PRINT "       TABELLA CODICI"
520 PRINT
530 PRINT"-COD--CAR----COD--CAR--"
535 FOR K = 0 TO 31
540 PRINT "  ";R+K,CHR$(R+K);"   ";R+K+1,"   ";
         CHR$(R+K+1)
545 LET K=K+1
```

```
550 NEXT K
560 PRINT "CAMBIO PAGINA (S/N)?"
570 INPUT R$
580 IF R$="N" THEN GO TO 620
590 IF R=1 THEN GO TO 605
592 LET R=1
600 GO TO 505
605 LET R=33
610 GO TO 505
620 CLS
630 GO TO 15
```

Note al programma:

La routine che visualizza i caratteri va dalla linea 40 alla linea 160. Se si risponde con un numero minore di zero si ottiene la tabella dei codici e dei caratteri. L'istruzione 120 da' come valori i numeri 0 e -1 a seconda che la condizione (V AND P(K))>0 sia falsa o vera.

Codifica di un programma per lo ZX81 e ZX80-Nuova ROM:

```
  1 CLS
  5 LET I = 0
 10 PRINT"SCRIVI IL CODICE"
 20 PRINT"<=63 O >=128 E <=191"
 30 INPUT X
 32 IF X>63 AND X<128 OR X>191 THEN GO TO 30
 33 PRINT AT 3,10;CHR$ 5;CHR$ X;CHR$ 133
 35 IF X<128 THEN GO TO 50
 40 LET I = 1
 45 LET X = X - 128
 50 FOR K=0 TO 7
 55 LET A$=""
 60 LET A=PEEK(7680+K+8*X)
 70 FOR V-0 TO 7
 75 LET R=A-INT(A/2)*2
 80 LET A$=CHR$(128*ABS(I-R))+A$
 85 LET A=INT(A/2)
 90 NEXT V
 92 PRINT AT K+3,1;A$
 93 PAUSE 20
 95 NEXT K
100 PRINT AT 21,0;"PREMI UN TASTO"
110 PAUSE 1000
120 CLS
122 RUN
```

Note al programma:

La routine di visualizzazione dei caratteri va dalla linea 33 alla linea 92.

9.14. COME RISOLVERE IL PROBLEMA DEI FILE DI DATI

Le due implementazioni del Basic disponibili sui calcolatori Sinclair non consentono di gestire file di dati in modo diretto. E' pero' possibile organizzare programmi che incorporino dati, record con i relativi campi, in apposite variabili del programma stesso. Al momento del SAVE del programma su cassetta, vengono salvate anche le variabili con i loro contenuti. Naturalmente dopo aver eseguito il LOAD di un programma contenente dei dati non si puo' farlo partire con il comando RUN, ma si deve mandarlo in esecuzione con GOTO N. Per non dimenticare di salvare il programma alla fine dell'esecuzione si puo' farlo terminare in modo opportuno e cioe' far comparire un messaggio che chieda di attaccare il registratore e di premere un tasto quando si e' pronti, e porre come ultima istruzione logica del programma una SAVE.

Questo si realizza per lo ZX80 con una sequenza del tipo:

```
9000 PRINT "MONTA IL NASTRO E AVVIA IL REGISTRATORE"
9001 PRINT "QUANDO SEI PRONTO PREMI NEW LINE"
9002 INPUT A$
9003 SAVE
```

e per lo ZX81 e lo ZX80-Nuova ROM con una sequenza del tipo:

```
9000 PRINT "MONTA IL NASTRO E AVVIA IL REGISTRATORE"
9001 PRINT "QUANDO SEI PRONTO PREMI UN TASTO"
9002 IF INKEY$ = "" THEN GOTO 9002
9003 SAVE "nome-programma"
```

Naturalmente la realizzazione di file di dati interni al programma e' piu' agevole nello ZX81 e ZX80-Nuova ROM dove sono disponibili le variabili stringa con indice, anche a dimensioni multiple. La cosa e' pero' realizzabile anche nello ZX80.

Si potrebbe anche ovviare al problema della partenza del programma con GOTO invece che con RUN, ma bisognerebbe usare il seguente artificio. Fare iniziare il programma con una serie di REM seguite da un numero fisso di caratteri, per esempio 50 REM seguite da 40 lineette. Ogni REM occupa lo stesso numero di caratteri in memoria e quindi conoscendo l'indirizzo di partenza del programma si puo' andare a scrivere con POKE e leggere con PEEK all'interno di ogni REM. Il programma va ugualmente salvato dopo ogni elaborazione, ma puo' sempre essere mandato in esecuzione con RUN.

Segue un esempio di questa tecnica per lo ZX80. Per brevita' scriviamo solo 10 REM, seguite da 40 lineette ciascuna; quello che importa e' impadronirsi del metodo.

```
  1 REM ----------------------------------------
  2 REM ----------------------------------------
  3 REM ----------------------------------------
  4 REM ----------------------------------------
  5 REM ----------------------------------------
  6 REM ----------------------------------------
  7 REM ----------------------------------------
  8 REM ----------------------------------------
  9 REM ----------------------------------------
 10 REM ----------------------------------------
500 LET M = 16424
510 LET N = 44
520 LET L = 3
525 LET C = 220
530 PRINT "SCRIVI IL DATO DA MEMORIZZARE"
540 INPUT A$
543 IF CODE(A$) = 1 THEN GOTO 650
545 LET I = 0
550 FOR K=1 TO 10
560 LET P=(K-1)*44+M+L
570 IF PEEK(P) = C THEN GOTO 770
580 NEXT K
590 PRINT "MANCA POSTO"
600 STOP
650 PRINT "TERMINATA MEMORIZZAZIONE"
660 PRINT "PREPARA REGISTRATORE E PREMI NEW LINE"
670 INPUT A$
690 SAVE
700 LET A=CODE(A$)
710 IF A = 1 THEN RETURN
720 IF I = 40 THEN RETURN
730 POKE P+I,A
740 LET A$ = TL$(A$)
750 LET I = I + 1
760 GOTO 700
770 GOSUB 700
780 LET K = 10
790 NEXT K
800 GOTO 530
```

Variabili usate nel programma:

. M contiene l'indirizzo inizio programma.

. N contiene la lunghezza di ogni REM in byte (2 per numero linea, 1 per REM, 40 per lineette, 1 per fine istruzione).

. L contiene la distanza della prima lineetta dall'inizio della istruzione REM.

. C contiene 220, codice della lineetta nella REM.
. A$ contiene il dato da memorizzare o il tasto premuto per i controlli.
. I e' il contatore dei caratteri che si possono memorizzare in ogni REM, al massimo 40.
. K e' la variabile di controllo del ciclo delle 10 possibili memorizzazioni.
. P e' il puntatore all'inizio della prima lineetta nella prima REM libera.
. A contiene il codice del carattere di A$ da memorizzare.

Note al programma:

. Le prime 10 linee sono le REM di memorizzazione
. Da 500 a 525 sono inizializzate le variabili per gestire la memorizzazione.
. Da 530 a 543 chiede il dato; se esso e' la stringa nulla va alla fase di rimemorizzazione del programma.
. Da 545 a 600 cerca il posto per memorizzare nelle 10 REM, se lo trova va alla linea 770, se no segnala che non c'e' piu' posto.
. Da 650 a 690 prepara ed esegue la memorizzazione.
. Da 700 a 760 si ha la routine di memorizzazione della stringa A$ troncando i caratteri se superano i 40.
. Da 770 a 800 c'e' la procedura per andare alla routine di memorizzazione se c'e' posto libero nelle REM.

Questo vuole solo essere un esempio; per gestirsi una procedura, come agenda indirizzi o simili, si devono scrivere altri programmi basati sempre su questa tecnica.
Se si desidera trasformare il programma per ZX81 o ZX80-Nuova ROM, si devono modificare alcune costanti di gestione delle REM e modificare la routine di memorizzazione del dato da 700 a 760. Inoltre per la sistemazione del registratore si puo' usare la INKEY$. Infatti con la nuova ROM il programma inizia a 16509, e la prima lineetta nelle REM e' dopo 5 byte.

La tecnica ora discussa e' poi la medesima suggerita nel Capitolo 8 per la memorizzazione di programmi in linguaggio macchina. Se il programma lo si fa iniziare, invece che con delle REM, con delle PRINT "caratteri", si devono fare i conti considerando i due byte in piu' per gli apici delimitatori; pero' quando si da' il RUN al programma si ottiene la lista di tutti i dati sul video e si puo' avere una fermata per schermo pieno se le righe sono molte.

Anche sullo ZX80 si puo' fare uso di stringhe per memorizzare dati all'interno di un programma, ma si urta con la limitazione delle possibili 26 stringhe. Le stringhe

possono essere lunghe a piacere e non devono essere tutte lunghe uguali. Per trattare stringhe di lunghezza variabile diventa piu' complicata la programmazione; si deve infatti definire un carattere delimitatore dei campi ed andarlo a ricercare quando si prelevano i diversi campi. Inoltre la scansione delle stringhe puo' essere fatta solo partendo dal primo carattere.

Per poter trattare parecchi dati, qualunque sia il metodo seguito, e' necessario disporre della espansione di memoria, sia per lo ZX80, che per i modelli con Nuova ROM.

Inoltre si raccomandano alcune cautele per non distruggere gli archivi. La prima regola e' di conservare sempre almeno una copia del vecchio archivio ogni volta che si fa un aggiornamento; cioe' di cambiare il nastro sul registratore e di apporvi una etichetta.

Quando si vuole gestire un archivio di . dati necessitano diversi programmi o un programma solo che faccia parecchie cose; le procedure necessarie sono le seguenti:

- creazione dell'archivio;
- aggiornamento dell'archivio: - correzione dati;
  - aggiunta dati;
  - cancellazione dati:
- eventuale ordinamento dei dati in base ad una chiave (l'archivio puo' essere creato gia' con un ordine);
- ricerca sull'archivio.

Nel caso dei calcolatori Sinclair qualunque tipo di archivio viene gestito tutto in memoria, si ha cioe' il tempo iniziale di caricamento dal nastro e poi la memoria e' tutta accessibile nello stesso tempo. Naturalmente, se l'archivio e' abbastanza grande, puo' essere importante il metodo di gestione ai fini del risparmio del tempo. Pensate, per esempio, di aver memorizzato 2000 nomi, tutti lunghi uguale, in ordine alfabetico. Un programma che li analizzi partendo dal primo impiega piu' tempo di un programma che faccia una ricerca binaria, cioe' che consideri l'elemento mediano e proceda continuando a dimezzare la tabella restringendo il campo di ricerca. Un'altra tecnica puo' essere quella di affiancare all'archivio una tabella di indici che conservi i puntatori al primo elemento che inizia con una lettera dell'alfabeto.

Con la Nuova ROM il trattamento degli archivi di dati risulta piu' semplice; infatti si possono memorizzare i diversi campi che costituiscono un record in una serie di vettori paralleli, aventi le stesse dimensioni.

Per creare una agenda di 100 indirizzi si possono definire

i seguenti vettori:

```
10 DIM C$(100,20)        cognome di 20 car.
11 DIM N$(100,15)        nome di 15 car.
12 DIM I$(100,25)        indirizzo di 25 car.
13 DIM L$(100,15)        localita' di 15 car.
14 DIM T$(100,12)        telefono di 12 car.
```

in ogni vettore a parita' di indice sono memorizzati i dati della stessa persona. Se il programma di creazione del file chiede i nominativi in ordine alfabetico e controlla ogni nominativo con quello ricevuto precedentemente, scartandolo se non e' in ordine, si ottiene una registrazione in ordine alfabetico. Naturalmente in questo caso la inserzione di nuovi nominativi comporta lo spostamento di tutti gli altri per liberare spazio; un problema simile si presenta con la cancellazione.

Riportiamo un esempio, ancora per lo ZX81 e ZX80-Nuova ROM, di memorizzazione di dati di lunghezza variabile che sfrutta la tecnica dei puntatori per reperire i dati. I dati sono memorizzati in una stringa unica D$, la quale viene utilizzata con la tecnica dello "slicing". Tale stringa viene iniziamente dimensionata a N caratteri per fissarla in memoria e riempirla di spazi. Si usa un vettore P, dimensionato a M+1 posizioni, per trattare M dati. E' responsabilita' dell'utente dare per N ed M dei valori congruenti.

```
 10 PRINT"SCRIVI DIMENSIONE STRINGA D$"
 15 INPUT N
 20 PRINT"SCRIVI QUANTI DATI"
 25 INPUT M
 30 DIM D$(N)
 40 DIM P(M+1)
 50 LET K=1
 55 PRINT"SCRIVI M PAROLE"
 60 FOR I=1 TO M
 65 PRINT TAB 5;I;") ";
 70 INPUT A$
 75 IF A$ =""THEN GO TO 70
 80 LET P(I)=K
 85 LET L=LEN(A$)
 90 LET D$(KTOK+L-1)=A$
 95 LET K=K+L
100 PRINT A$
105 NEXT I
110 LET P(I)=K
115 PAUSE 200
120 POKE 16437,255
140 CLS
145 PRINT AT 10,0;"SCRIVI NUMERO DEL DATO, 0 PER USCIRE"
```

```
150 INPUT X
155 IF X=0 THEN STOP
160 IF X<0 OR X>M THEN GOTO 150
165 CLS
170 PRINT AT 10,7;"NUMERO DATO ";X
175 PRINT AT 12,14;"DATO"
180 PRINT AT 14,(31+P(X)-P(X+1))/2;D$(P(X)TOP(X+1)-1)
185 GOTO 115
```

Variabili usate nel programma:

- N dimensioni stringa D$ in caratteri;
- M numero dei dati da memorizzare;
- K puntatore all'inizio dato entro la stringa;
- I contatore ciclo memorizzazione dati;
- D$ stringa per i dati di N caratteri;
- P vettore dei puntatori ai dati;
- A$ stringa per leggere il dato;
- L lunghezza dato letto;
- X numero dato da listare.

Note al programma:

- Da 10 a 40 vengono precisate le dimensioni dei dati.
- Da 50 a 105 vengono letti e memorizzati i dati aggiornando i puntatori nel vettore P. Da notare l'uso dello "slicing" all'interno della stringa D$.
- Nella 110 viene memorizzato il tappo nella posizione in piu' del vettore dei puntatori.
- Da 115 a 140 si ha la pausa, poi il ripristino necessario per il modo FAST dello ZX81 e lo ZX80-Nuova ROM, e la pulizia dello schermo.
- Da 145 a 185 si ha la richiesta di dato e la stampa sullo schermo del dato; se X = 0 il programma termina. Nella linea 180 si usa una formuletta per fare apparire il dato centrato sul video.

Dall'esempio precedente potete ricavare delle idee per i vostri programmi.

9.15. IL GIOCO DELLE SFERE SU ZX81 E ZX80-Nuova ROM

Questo gioco consiste nell'indovinare il volume di una sfera, dato il suo diametro, con una approssimazione di piu' o meno 0.5.

All'inizio viene chiesto al giocatore di quante cifre si compone il numero che rappresenta il diametro della sfera (per risposta 2, potra' venire proposto come diametro un numero intero da 0 a 99). Il programma evidenzia sul video il numero delle cifre del diametro, disegna la sfera e

scrive la misura del diametro. Poi viene richiesta la misura del volume della sfera. Se il giocatore risponde in modo esatto il programma lo conferma, altrimenti viene mostrato quale avrebbe dovuto essere la risposta. Per continuare a giocare si deve rispondere con S, per fermare il programma con qualunque altro tasto, sempre seguito da NEW LINE.

Codifica del programma:

```
 10 RAND
 20 CLS
 30 PRINT "NUMERO CIFRE ?"
 40 INPUT A
 50 SCROLL
 60 PRINT AT 0,0;"NUMERO CIFRE ";A
 70 FOR N=0 TO 20
 80 PLOT 5+4*SIN(N*PI/10), 10+4*COS(N*PI/10)
 90 NEXT N
100 PRINT AT 20,0;"IL DIAMETRO MISURA ";
110 LET B=INT(RND*10**A)
120 PRINT B
140 PRINT "VOLUME DELLA SFERA ?"
150 INPUT C
160 SCROLL
170 SCROLL
175 LET V = PI*B**3/6
180 IF ABS(C-V)> = .5 THEN PRINT C;" RISULTA ERRATO"
190 SCROLL
200 SCROLL
210 PRINT INT(V + .5);" RISULTATO ESATTO"
230 INPUT A$
240 IF A$ = "S" THEN RUN
```

Variabili usate nel programma:

- A numero massimo di cifre del diametro.
- B diametro proposto dal programma.
- C volume sfera proposto dal giocatore.
- V volume sfera calcolato dal programma.
- N variabile controllo ciclo disegno sfera.
- A$ risposta giocatore.

Note al programma:

. Da 10 a 20 si predispone la partenza per estrarre i numeri a caso e si pulisce lo schermo.

. Da 30 a 50 viene chiesto il numero delle cifre per il diametro.

. La 60 scrive in alto a sinistra il livello delle cifre.

. Da 70 a 90 viene disegnata la sfera.
. Da 100 a 120 viene estratto a caso il diametro nell'ambito del livello di cifre scelto.
. Da 140 a 170 viene richiesto il volume al giocatore e memorizzata la risposta e preparato spazio sul video.
. A 175 viene calcolato il volume V.
. Da 180 a 210 si ha il controllo della risposta del giocatore e la stampa dei risultati.
. Da 230 a 240 si ha il colloquio per decidere se continuare il gioco.

Provate il programma anche in modo SLOW sullo ZX81.

9.16. L'ANIMAZIONE DELLE FIGURE SULLO ZX81

Quando lo ZX81 funziona in modo SLOW si e' nelle condizioni ideali per realizzare l'animazione delle figure sullo schermo. Infatti lo schermo non viene cancellato mentre il calcolatore lavora. L'animazione si ottiene spostando una figura in diverse posizioni del video e facendola permanere in ogni posizione per un certo tempo. Variando il tempo si ottiene una maggiore o minore velocita' del movimento.

Lo spostamento delle figure si ottiene con la funzione AT o con i comandi PLOT e UNPLOT. La funzione AT consente di scrivere qualunque carattere in una delle posizioni dello schermo con risoluzione pari alle dimensioni di un carattere. Il comando PLOT aumenta la risoluzione e consente di scrivere un quadratino nero pari ad un quarto del cursore in una posizione delle 64 colonne e 44 righe dello schermo (raddoppiando il numero delle colonne e delle righe). Per il comando PLOT la posizione di coordinate 0,0 si trova nell'angolo in basso a sinistra dello schermo; l'asse delle X e' orizzontale e quella delle Y verticale. Per la funzione AT la posizione di coordinate 0,0 si trova nell'angolo in alto a sinistra del video; l'asse delle X e' verticale orientata verso il basso e l'asse delle Y e' orizzontale orientata verso destra. Si ha cioe', rispetto alla PLOT uno spostamento dell'origine degli assi ed una rotazione di 90 gradi.
Con il programma che segue, usando i comandi PLOT e UNPLOT, si ottiene di vedere muovere il quadratino nero lungo i bordi del video in senso antiorario.

```
 3 PRINT"SCRIVI IL TEMPO DEL MOVIMENTO"
 5 INPUT N
 7 CLS
10 LET Y=0
```

```
 20 LET X1=0
 30 LET X2=63
 35 LET N1=1
 40 GOSUB 200
 50 LET X=63
 60 LET Y1=0
 70 LET Y2=43
 75 LET N1=1
 80 GOSUB 300
 90 LET Y=43
100 LET X1=63
110 LET X2=0
115 LET N1=-1
120 GOSUB 200
130 LET X=0
140 LET Y1=43
150 LET Y2=0
155 LET N1=-1
160 GOSUB 300
170 PRINT AT 10,10;"FINITO"
180 STOP
200 FOR X=X1 TO X2 STEP N1
210 PLOT X,Y
215 PAUSE N
220 POKE 16437,255
230 UNPLOT X,Y
240 NEXT X
250 RETURN
300 FOR Y=Y1 TO Y2 STEP N1
310 PLOT X,Y
315 PAUSE N
320 POKE 16437,255
330 UNPLOT X,Y
340 NEXT Y
350 RETURN
```

Note al programma:

. Si usano due sottoprogrammi, 200 e 300, per ottenere il movimento orizzontale o verticale. All'inizio viene chiesto l'intervallo N per la pausa; provate con diversi valori di N. La linea 320 non sarebbe necessaria per lo ZX81 in modo SLOW, mentre diventa necessaria in modo FAST.

Si possono far muovere oggetti come nel programma che segue, usando la tecnica di cancellare con UNPLOT solo una parte dell'oggetto.

```
 3 PRINT"SCRIVI IL TEMPO DEL MOVIMENTO"
 5 INPUT N
10 FOR X=0 TO 61
```

```
15 PLOT X,0
20 PLOT X+1,0
25 PLOT X+2,0
30 PLOT X+1,1
35 PAUSE N
40 POKE 16437,255
45 UNPLOT X,0
50 UNPLOT X+1,1
55 NEXT X
```

Note al programma:

. All'inizio viene chiesto N, il tempo della pausa. Viene fabbricato un oggetto nell'angolo sinistro in basso dello schermo e lo si fa muovere fino all'angolo destro in basso.

Per far muovere gli oggetti a comando si possono usare i tasti di comando dei cursori dello schermo (SHIFT + 5 cursore a sinistra, SHIFT +8 cursore a destra, SHIFT + 6 puntatore linea in basso, SHIFT + 7 puntatore linea in alto) sfruttando la funzione INKEY$. Non si usano quei tasti perche' agiscono direttamente sul video, ma solo perche' sono mnemonici per l'utente in quanto riportano le frecce relative ai 4 tipi di movimento possibili; il programma modifica le coordinate della posizione del video controllando il tasto premuto.

Nel programma che segue e' riportato un esempio di questa tecnica.

```
10 LET X=10
20 LET Y=15
30 LET A$=CHR$ 8
40 IF INKEY$ ="5" THEN LET Y=Y-1
50 IF INKEY$ ="6" THEN LET X=X+1
60 IF INKEY$ ="7" THEN LET X=X-1
70 IF INKEY$ ="8" THEN LET Y=Y+1
80 PRINT AT X,Y;A$
90 GOTO 40
```

Il carattere grafico viene solo spostato, ma non si cancella il precedente.

Riportiamo un semplice gioco che sfrutta le capacita' grafiche sopra esaminate. Le regole sono le seguenti: mentre ad intervalli di tempo prefissati una barrettina verticale chiara si muove lungo la striscia nera tracciata sotto le caselle dei numeri da 1 a 9 il giocatore deve premere il tasto del numero sotto il quale essa passa. Se il momento della pressione del tasto coincide con il passaggio della

barrettina il giocatore ha colpito e guadagna un punto. La scansione della striscia viene fatta 10 volte.

```
  5 LET S$=CHR$128+CHR$128+CHR$128
  7 LET T$=CHR$128+CHR$128
 10 LET K = 0
 20 PRINT
 25 PRINTS$;"1";T$;"2";T$;"3";T$;"4";T$;"5";
 26 PRINTT$;"6";T$;"7";T$;"8";T$;"9";S$
 30 PRINT AT 4,0;"SEI A"
 40 PRINT AT 4,22;"COLPITO"
 45 PRINT AT 4,10;"INIZIA    "
 50 FOR I=1 TO 10
 60 LET P=0
 65 PRINT AT 3,0;S$;S$;S$;S$;S$;S$;S$;S$;S$;S$
 70 PRINT AT 4,6;I;TAB 10;"           "
 75 PRINT AT 3,P;CHR$128;CHR$5
 80 LET N=CODE INKEY$-28
 85 IF N<1 OR N>9 THEN GO TO 130
 90 PRINT AT 3,N*3;CHR$151
 95 IF N*3<>P+1 THEN GOTO 125
100 PRINT AT 4,12;"COLPITO"
110 LET K=K+1
115 PRINT AT 4,30;K
120 GO TO 140
125 IF INKEY$<>"" THEN GOTO 125
130 LET P=P+1
135 IF P<>31 THEN GOTO 75
140 PAUSE 250
150 NEXT I
155 PRINT AT 4,10;"FINE GIOCO"
```

Variabili usate nel programma:

- S$ per contenere 3 spazi.
- T$ per contenere 2 spazi.
- K per contare i punti del giocatore.
- I per controllare il ciclo delle 10 scansioni.
- N per calcolare il tasto premuto.
- P per contare gli spostamenti della barretta.

Note al programma:

. Non e' significativo usare il programma in modo FAST.

. Nelle linee 5 e 7 si preparano delle variabili contenenti spazi; si poteva ottenere lo stesso risultato usando gli spazi inversi tra apici invece di CHR$128.

. La 10 azzera il contatore K dei colpi andati a segno.

. Dalla 20 alla 40 viene preparato il tracciato del gioco sul video.

. Nella 45 viene avvisato il giocatore che inizia il gioco.

. Dalla 50 alla 150 c'e' il ciclo di 10 scansioni della serie di numeri. Nella 60 viene posto a zero il contatore P delle 31 posizioni della striscia nera che viene percorsa dalla barretta verticale (CHR$ 5). Nella 65 viene tracciata la striscia nera sotto le caselle dei numeri. Nella 75 viene posizionata la barretta sulla striscia in base al valore di P. Nella 80 viene posto N al codice del tasto schiacciato - 28; questo per ottenere dai tasti numerici le cifre da 1 a 9; se non e' stato premuto alcun tasto N e' fuori dall'intervallo 1 - 9. Se N e' fuori dall'intervallo 1 - 9 la 85 manda alla 130, dove P viene incrementato di 1. Alla 135 viene analizzato P, se non e' uguale a 31 si torna alla 75 e la barrettina prosegue il suo cammino, se P = 31 si ha con la 140 una pausa; al termine della pausa con la 150 NEXT I si torna alla 60 se non e' terminato il ciclo di 10 scansioni, altrimenti il gioco termina con la linea 155. Se alla 85 N viene trovato dentro l'intervallo, alla 90 viene stampato un asterisco in campo inverso (CHR$ 151) in corrispondenza del numero giocato. Alla 95 viene analizzata la posizione dove e' stato messo l'asterisco confrontandola con il valore attuale del contatore di posizione P; se si ha coincidenza il giocatore ha colpito e guadagna 1 punto e questo viene segnalato dalla 100 alla 115, dopo di che si va alla pausa della 140. Cioe' il giocatore puo' colpire una sola volta durante una scansione. Se alla 95 risulta che il giocatore non ha colpito si va alla 125 dove con l'uso della INKEY$ viene scartata un'altra eventuale pressione di tasto e poi viene incrementato P alla 130.

9.17. LO ZX81 DISEGNA

Potete ottenere dei disegni apportando poche modifiche ai primi due programmi esempio del precedente paragrafo. Nel primo cancellate le linee: 215, 220, 230, 315, 320, 330 e poi provate il programma. Vedrete un bordo nero attorno al video. Se nel secondo togliete le linee: 35, 40, 45, 50 e provate vedrete un disegno.
Provate i programmi sia nel modo FAST che nel modo SLOW.

Proviamo ora a disegnare un cerchio fornendo il centro ed il raggio.

```
10 PRINT "SCRIVI COORDINATE A E B DEL CENTRO"
15 PRINT "E IL RAGGIO R"
20 CLS
25 PRINT "CENTRO: A =          B ="
30 INPUT A
35 INPUT B
37 PRINT AT 0,12;A;AT 0,25;B
```

```
39 PRINT "RAGGIO="
40 INPUT R
45 PRINT AT 1,8;R
50 FOR K=0 TO 360
55 LET Z=K*PI/180
60 PLOT A+R*COS Z,B+R*SIN Z
65 NEXT K
```

Note al programma:

. Non vengono controllati i valori per la congruenza con le dimensioni del video.

. Si usano le formule: X=A+R*COS Z e Y=B+R*SIN Z.

. Il disegno si sviluppa lentamente perche' il calcolatore impiega un po' di tempo nel calcolo delle funzioni trigonometriche.

Per disegnare una parabola con la concavita' rivolta verso il basso potete provare questo programma:

```
100 FOR X=0 TO 63
105 LET Y=INT((2.52-0.04*X)*X)
110 PLOT X,Y
115 NEXT X
```

Per disegnare una ellisse:

```
100 PRINT "SCRIVI I PARAMETRI A E B DELLA ELLISSE"
105 PRINT "A = ";
110 INPUT A
115 PRINT A;"   B = ";
120 INPUT B
125 PRINT B
127 CLS
130 FOR K=0 TO 360
135 LET Z=K*PI/180
140 PLOT A*(1+COS Z),B*(1+SIN Z)
145 NEXT K
```

provate dando per A e B diversi valori come: (2,21), (20,20), (30,20).

Segue un programma che genera disegni casuali usando un carattere scelto dall'utente:

```
 3 PRINT "SCRIVI IL TEMPO BASE"
 5 INPUT T
10 PRINT AT 5,8;"DISEGNI CASUALI"
15 PRINT AT 10,0;"SCRIVI IL CARATTERE CHE VUOI USARE"
```

```
 20 INPUT A$
 25 CLS
 30 FOR I = 1 TO 600
 35 LET X=INT(RND*11)+1
 40 LET Y=INT(RND*16)+1
 45 PRINT AT X,Y;A$
 50 PAUSE T
 55 PRINT AT 22-X,Y;A$
 60 PAUSE T
 65 PRINT AT X,32-Y;A$
 70 PAUSE T
 75 PRINT AT 22-X,32-Y;A$
 80 PAUSE 3*T
 85 LET X=INT(RND*11)+1
 90 LET Y=INT(RND*16)+1
 95 PRINT AT X,Y;"*"
100 PAUSE T/6
105 PRINT AT 22-X,Y;"*"
110 PAUSE T/6
115 PRINT AT X,32-Y;"*"
120 PAUSE T/6
125 PRINT AT 22-X,32-Y;"*"
130 PAUSE T/6
135 IF RND>.4THEN GOTO 85
140 NEXT I
145 PAUSE 12*T
150 CLS
155 RUN
```

Variabili usate nel programma:

- T tempo base per le pause.
- A$ carattere per disegnare.
- I variabile di controllo del ciclo.
- X e Y coordinate del punto dove disegnare.

Note al programma:

. Dalla 3 alla 25 viene chiesto il tempo base T per le pause, il carattere da usare nel disegno e viene pulito lo schermo.

. Dalla 30 alla 140 si ha il ciclo di disegno, qui programmato 600 volte. La X e la Y, coordinate del punto dove disegnare il carattere scelto, vengono calcolate usando la RND in modo che sia X<=12 e Y<=17 per il primo carattere, poi ne vengono disegnati altri 3 in posizioni simmetriche. Tra il disegno di un carattere ed il successivo si ha una pausa. Dopo una pausa tripla delle precedenti vengono disegnati con lo stesso metodo 4 asterischi, solo che la pausa tra l'uno e l'altro e' piu' breve. Si ottiene un gradevole disegno sul video, sempre diverso, dato che viene sfruttata la RND. La linea 135 analizza un numero a caso e

se esso e' < 0.4 torna a disegnare asterischi invece del carattere di A$.

. Terminato il ciclo si ha una pausa piuttusto lunga, poi viene pulito il video ed il programma ricomincia, infatti la linea 155 rida' il RUN.

Potete modificare il numero 600 in uno minore per non fare durare troppo il programma e potete modificare il tempo base T per ottenere un effetto diverso.

9.18. ANIMAZIONE E DISEGNI PER LO ZX80-NUOVA ROM

Lo ZX80-Nuova ROM differisce dallo ZX81 per il fatto che il tasto FAST e il tasto SLOW non sono attivi da tastiera. Questo comporta che se si vuole ottenere l'animazione delle figure si deve ricorrere a qualche artificio di programmazione tipo ZX80 per ottenere la miglior persistenza delle immagini sullo schermo. Il comportamento di questo calcolatore e' assolutamente uguale a quello dello ZX81 funzionante in modo FAST. Potete provare a trasformare i programmi di animazione dello ZX80, con le dovute modifiche, per lo ZX80-Nuova ROM.

Ricordate che se usate il comando PAUSE, dovete farlo seguire da POKE 16437,255.

Per quanto riguarda i disegni valgono le stesse osservazioni fatte a proposito dell'animazione.

9.19. IL GIOCO DELLA SPIRALE SULLO ZX80

Questo programma puo' essere provato sullo ZX80 con 1K di memoria. Il gioco consiste nel muoversi all'interno dei corridoi della spirale, arrivando nel piu' breve tempo possibile al centro. Il centro e' l'ultimo quadratino nero dove termina il bordo della spirale nella posizione centrale. I bordi della spirale sono ottenuti usando il carattere CHR$(128), che e' lo spazio in campo inverso, mentre i corridoi sono ottenuti con lo spazio. La pedina del giocatore e' contraddistinta dal segno +. Il giocatore deve muoversi senza andare sui bordi; se ci va deve tornare nel corridoio facendo la mossa inversa alla precedente. Per muovere la pedina il giocatore deve usare i quattro tasti che recano le frecce di movimento cursore; in tale modo ha un riferimento mnemonico alle mosse che vuole fare. Il significato dei tasti e' il seguente:

. 5-freccia a sinistra, per andare a sinistra;
. 6-freccia in basso, per andare verso il basso;

. 7-freccia in alto, per andare verso l'alto;
. 8-freccia a destra, per andare a destra.

Il giocatore puo' muovere quando appare il cursore sdoppiato con LS per richiesta di INPUT numerico sotto il punteggio attuale. Il punteggio parte da un valore alto (9999) e viene scalato in base al tempo che il giocatore impiega per ogni mossa. Il tempo impiegato per ogni mossa viene calcolato in base al contatore dei fotogrammi dello schermo.

La codifica del programma e' la seguente:

```
  3 LET E$=CHR$(128)
  5 LET P$=CHR$(19)
 10 PRINT "  SPIRALE "
 20 POKE 16421,24
 30 PRINT
 40 PRINT E$;E$;E$;E$;E$;E$;E$;E$;E$;E$;E$
 50 PRINT E$;P$;E$;"8 spazi"
 60 PRINT E$;"1 spazio";E$;"1 spazio";E$;E$;E$;E$;E$;
          "1 spazio";E$
 70 PRINT E$;"1 spazio";E$;"1 spazio";E$;"3 spazi";E$;
          "1 spazio";E$
 80 PRINT E$;"1 spazio";E$;"1 spazio";E$;"1 spazio";E$;
          "1 spazio";E$;"1 spazio";E$
 90 PRINT E$;"1 spazio";E$;"1 spazio";E$;E$;E$;
          "1 spazio";E$;"1 spazio";E$
 95 PRINT E$;"1 spazio";E$;"5 spazi";E$;"1 spazio";E$
100 PRINT E$;"1 spazio";E$;E$;E$;E$;E$;E$;E$;"1 spazio";
          E$
110 PRINT E$;"9 spazi";E$
120 PRINT E$;E$;E$;E$;E$;E$;E$;E$;E$;E$;E$
130 PRINT
140 PRINT "TUO PUNTEGGIO 9999"
150 POKE 16414,0
160 POKE 16415,0
170 LET X=26
180 LET A=0
190 IF A=66 THEN GOTO 530
200 INPUT N
210 IF N=5 THEN LET Y=X-1
220 IF N=6 THEN LET Y=X+12
230 IF N=7 THEN LET Y=X-12
240 IF N=8 THEN LET Y=X+1
250 IF A AND Y-A THEN GOTO 430
260 IF A THEN GOTO 350
270 LET U=X
280 LET V=0
290 GOSUB 510
300 IF PEEK(PEEK(16396)+PEEK(16397)*256+Y)-128 THEN
    GOTO 390
310 LET A=X
```

```
320 LET U=Y
330 LET V=147
335 GOSUB 510
340 GOTO 420
350 LET A=0
360 LET U=X
370 LET V=128
380 GOSUB 510
390 LET U=Y
400 LET V=19
410 GOSUB 510
420 LET X=Y
430 LET S$=STR$(9999-PEEK(16414)-PEEK(16415)*256)
440 FOR J=1 TO 4
450 LET U=147+J
460 LET V=CODE(S$)
470 GOSUB 510
480 LET S$=TL$(S$)
490 NEXT J
500 GOTO 190
510 POKE PEEK(16396)+PEEK(16397)*256 + U,V
520 RETURN
530 PRINT
540 PRINT "FINE GIOCO"
```

Variabili usate nel programma:

- E$ = CHR$(128) spazio in campo inverso.
- P$ = CHR$(19) segno +, pedina giocatore.
- X posizione della pedina rispetto all'inizio della memoria di schermo.
- A valore per il controllo della posizione finale.
- Y posizione della pedina rispetto all'inizio della memoria di schermo in base all'ultima mossa.
- N valore della mossa (5,6,7,8).
- V codice ASCII del carattere da scrivere sul video.
- U spostamento dall'inizio della memoria di schermo per andare a segnare la mossa.
- S$ stringa contenente il punteggio aggiornato.
- J variabile di controllo del ciclo di aggiornamento del punteggio.

. I 2 byte 16397 e 16396 contengono l'indirizzo di inizio della memoria di schermo; in dipendenza dalle tecniche di programmazione usate, tale indirizzo resta costante durante l'esecuzione del programma.

. I 2 byte 16414 e 16415 contengono il numero dei fotogrammi dello schermo; essi vengono azzerati all'inizio del gioco.

. Il byte 16421 contiene la posizione di riga del cursore; esso viene posto alla linea 24, cioe' la prima al di sopra del video.

Note al programma:

. Da 10 a 140 viene disegnata sul video la spirale e viene lasciato il cursore prima del video (linea 24).
. Da 150 a 160 si azzerano i byte del contatore fotogrammi.
. Da 170 a 180 si inizializzano le variabili.
. La 190 controlla se il gioco e' finito.
. Da 200 a 260 si ha la richiesta della mossa e il controllo della stessa. Vengono calcolate le coordinate della mossa ed eseguito il sottoprogramma di aggiornamento del punteggio.
. Da 270 a 290 viene cancellata la vecchia mossa.
. La 300 controlla se la posizione raggiunta e' di bordo. Se non lo e' il programma prosegue dalla 390.
. Da 310 a 340 evidenzia un + in campo inverso sulla posizione di bordo raggiunta e prosegue da 420.
. Da 350 a 380 ripulisce il bordo dalla mossa errata.
. Da 390 a 410 viene segnata la mossa esatta.
. A 420 viene aggiornato X, variabile di posizione pedina.
. Da 430 a 500 viene aggiornato il punteggio lavorando sulla stringa. Il tempo viene calcolato in base al contatore dei fotogrammi. Il programma prosegue da 190.
. Da 510 a 520 viene scritto il carattere V nella posizione U del video.
. Da 530 a 540 viene segnalata la fine del gioco.
. Si raccomanda di scrivere il programma con cura onde evitare di dimenticare qualche spazio essenziale; essi sono stati indicati segnando un numero seguito dalla parola spazio tra apici.

Provate a modificare il programma per poterlo usare sugli altri modelli Sinclair.

9.20. FACCIAMO CENTRO SULLO ZX81 E ZX80-NUOVA ROM

In questo paragrafo sono riportati 2 programmi; il secondo e' un ampliamento del primo.

Il primo programma consiste nel centrare un canestro con una pallina. La pallina inizialmente si trova in alto a sinistra e si muove verso il basso potendo fare 11 movimenti di lunghezza crescente. Essa inizialmente non ha movimenti orizzontali e quindi la sua velocita' nella direzione orizzontale e' nulla. Il giocatore puo' premere il tasto 8 per imprimere una velocita' di spostamento orizzontale alla pallina da sinistra a destra, egli deve cercare di controbilanciare gli spostamenti verticali per fare arrivare

la pallina nel canestro che si trova in basso. La posizione del canestro in basso e' casuale, ma esso rimane fermo durante il gioco. L'impulso di spostamento orizzontale dipende dal numero delle pressioni sul tasto 8 e dal tempo della pressione; se l'impulso e' esagerato un "OOPS" segnala che il giocatore e' uscito dallo schermo. Se il canestro (rappresentato da "(-)") viene centrato si vede comparire un "** WOW **" di incoraggiamento.

Il programma si compone di 8 parti:

. 1) Posizionamento del canestro.
. 2) Disegno della pallina.
. 3) Eventuale modifica impulso orizzontale.
. 4) Calcolo nuova posizione pallina.
. 5) Se la pallina non e' arrivata in fondo allo schermo in basso ed e' ancora visibile ritorno al punto 2).
. 6) Se la pallina ha fatto centro messaggio "** WOW **".
. 7) Se la pallina e' sparita messaggio "OOPS" .
. 8) Richiesta di premere un tasto per ricominciare.

La codifica del programma e' la seguente:

```
  5 CLS
 10 LET P=INT(RND*26)+5
 20 PRINT AT 21,P-1;"(-)"
 30 LET P=P*2
 40 LET X=0
 50 PAUSE 20
 60 POKE 16437,255
 90 LET T=0
200 FOR Y=0 TO 11
210 IF INKEY$="8" THEN LET T=T+1
220 LET X=X+T
230 IF X>63 THEN GOTO 400
240 PLOT X,39-32*(Y/10)**2
250 IF INKEY$="8" THEN LET T=T+1
255 PAUSE 20
258 POKE 16437,255
260 NEXT Y
300 IF ABS(P-X)<=2 THEN PRINT AT 11,15;"** WOW **"
310 PRINT AT 0,0;"PREMI UN TASTO"
315 PAUSE 4E4
320 POKE 16437,255
330 RUN
400 PRINT AT 18,6;"OOPS"
410 GOTO 310
```

Variabili usate nel programma:

. P posizione del carattere centrale del canestro.

. X ascissa della pallina (scostamento dall'estremo sinistro).

. T velocita' orizzontale della pallina.

. Y contatore di ciclo, va da 0 a 11.

Note al programma:

. La 5 pulisce lo schermo.

. Da 10 a 20 disegno canestro.

. La 30 moltiplica P*2; si ricorda che la larghezza di un "pixel" e' la meta' di quella di un carattere.

. La 40 inizializza l'ascissa della pallina.

. La 200 inizia il ciclo per K da 0 a 11.

. La 210 controlla se il tasto 8 e' premuto; se si con T=T+1 incrementa la velocita' orizzontale della pallina.

. Da 220 a 230 calcola la nuova ascissa della pallina e se essa e' > 63 va alla 400.

. La 240 disegna la pallina.

. La 250 come la 210.

. Da 255 a 258 pausa per consentire la visualizzazione.

. La 260 rimanda a 210 se il ciclo di K e' terminato.

. Alla 300 la pallina e' arrivata al livello del canestro, se la sua distanza dal centro del canestrp e' < = 2 si ha il messaggio WOW.

. Da 310 a 320 crea una pausa in attesa che sia premuto un tasto.

. La 330 fa ripartire il programma.

. Da 400 a 410 scrive il messaggio OOPS e poi torna alla linea 310.

Se desiderate rendere piu' difficile il gioco potete sostituire nella linea 300 il <= 2 con <= 1.

Il secondo programma consiste nel far cadere un paracadutista su una zona assegnata. L'analogia con il programma precedente e' evidente, solo che questa volta non si fa uso della istruzione PLOT e non si lascia una scia.

Per ottenere una maggiore stabilita' dell'immagine si fa uso della PRINT AT.

Nel listato le linee da 10 a 16 definiscono delle stringhe contenenti i caratteri grafici necessari per disegnare il paracadutista. Volendo queste linee possono essere omesse e si devono preparare alle linee da 30 a 36 delle stringhe contenenti 8 spazi ciascuna e poi usando i relativi tasti andarci a disegnare dentro. Si riporta il contenuto delle linee da 30 a 36 ed anche da 320 a 330 ottenute in questo modo e listate con la stampante del Sinclair.

```
 6 RAND
 25 LET S$="
                    "
 30 LET A$="
 32 LET B$="
 34 LET C$="
 36 LET D$="
 50 LET S=0
 60 LET U=0
150 CLS
160 LET U=U+1
170 LET P=INT (RND*23)+3
180 PRINT AT 21,P-2;"     "
190 LET I=0
195 LET H=INT (RND*24)
```

```
315 PRINT AT 18,H;S$
320 PRINT AT 20,H;"*      *"
330 PRINT AT 21,H;"        "
340 PRINT AT 1,0;"PREMI NEW-LIN
E"
350 INPUT I$
360 LET S=S+10-ABS (H-P)
370 GOTO 150
400 PRINT AT 15,10;"**BEN ATTER
RATO**"
420 GOTO 340
```

Il gioco consiste nel guidare il paracadutista sulla piattaforma di atterraggio utilizzandi i tasti 5 (spostamento a sinistra) e 8 (spostamento a destra) che lo fanno spostare di 2 pixel e quindi di un carattere. Il povero paracadutista cade per effetto della gravita' ed inoltre casualmente tira vento e questo influisce sul suo movimento. Il vento ha una direzione indicata dal simbolo "<" o ">" in alto sullo schermo.

E' consigliabile sullo ZX81 provare il programma in FAST e poi in SLOW.

La codifica del programma e' la seguente:

```
  5 CLS
 10 LET T$=CHR$(135)
 11 LET U$=CHR$(3)
 12 LET V$=CHR$(134)
 13 LET W$=CHR$(1)
 14 LET X$=CHR$(132)
 15 LET Y$=CHR$(128)
 16 LET Z$=CHR$(0)
 25 LET S$="64 spazi"
 30 LET A$=Z$+Z$+T$+U$+V$+Z$+Z$+Z$
 32 LET B$=Z$+Z$+V$+T$+T$+W$+Z$+Z$
 34 LET C$=Z$+Z$+Z$+X$+W$+Z$+Z$+Z$
 36 LET D$=Z$+Z$+Z$+W$+W$+Z$+Z$+Z$
 50 LET S=0
 60 LET U=0
150 CLS
160 LET U=U+1
170 LET P=INT(RND*23)+3
180 PRINT AT 21,P-2;Y$+Y$+Y$+Y$+Y$
```

```
190 LET I=0
195 LET H=10
200 LET Y=0
206 IF I>2 THEN PRINT AT I-2,H;S$
208 LET V=SGN(1-RND*2)
209 PRINT AT 0,0;"VENTO:";CHR$(19-(V=1));
          Z$+Z$,"PUNTI:";S,"GIOCO:";U
210 PRINT AT I-1,H;"8 spazi"
212 PRINT AT I,H;A$;
215 PRINT TAB H;B$;
220 PRINT TAB H;C$;
225 PRINT TAB H;B$;
228 LET H=H+V
230 IF INKEY$="5" THEN LET H=H-1
234 IF INKEY$="8" THEN LET H=H+1
236 IF H>24 THEN LET H=24
238 IF H<0 THEN LET H=0
240 LET I=16*(Y/15)**2
245 IF Y=17 THEN GOTO 300
250 LET Y=Y+1
260 PAUSE 30
265 POKE 16437,255
270 GOTO 206
300 IF ABS(H-P+2)<3 THEN GOTO 400
315 PRINT AT 18,H;S$
320 PRINT AT 20,H;"* "+T$+T$+" *"
330 PRINT AT 21,H;Z$+T$+Y$+Y$+Y$+CHR$(130)
340 PRINT AT 1,0;"PREMI NEW LINE"
350 INPUT I$
360 LET S=S+10-ABS(H-P)
370 GOTO 150
400 PRINT AT 16,10;"**BEN ATTERRATO**"
420 GOTO 350
```

Variabili usate nel programma:

. T$, U$, V$, W$, X$, Y$, Z$, stringhe contenenti caratteri necessari per disegnare.

. S$ stringa usata per pulire lo schermo al di sopra del paracadutista.

. A$, B$, C$, D$, stringhe usate per disegnare il paracadutista.

. S punteggio.

. U numero partita (GIOCO).

. P posizione carattere centrale piattaforma.

. I altezza paracadutista (parte piu' alta).

. H ascissa del paracadutista (primo carattere).

. Y contatore: 17 significa paracadutista a livello terra.

. V direzione vento.

. I$ stringa di INPUT per far ripartire il programma.

Note al programma:

. Da 50 a 60 azzeramento contatore U e punteggio S.

. Da 150 a 200 viene pulito il video, e' calcolata in modo random la posizione della piattaforma ed essa viene disegnata, le variabili di controllo vengono azzerate.

. La 206 ripulisce due linee sopra il paracadutista se la sua distanza dall'alto supera 2.

. La 208 calcola la direzione del vento.

. Da 209 a 225 stampa l'intestazione dello schermo e disegna il paracadutista.

. Da 228 a 238 modifica l'ascissa del paracadutista in dipendenza dal vento e dall'INPUT da tastiera ed anche dalla distanza dal bordo dello schermo.

. La 240 calcola l'altezza del paracadutista con una formula diversa da quella usata nel precedente programma dato che si usa PRINT invece di PLOT.

. La 245 salta a 300 se il paracadutista ha toccatto terra.

. La 250 incrementa il contatore V.

. Da 260 a 270 si ha una pausa e poi va a 206.

. La 300 manda alla linea 400 se il paracadutista e' sopra la pedana.

. Da 315 a 330 disegna il paracadutista rotto.

. Da 340 a 370 aspetta NEW LINE per partire, calcola il punteggio e va alla linea 150.

. Da 400 a 420 scrive il messaggio di felice atterraggio e salta alla linea 340.

9.21. AGENDA TELEFONICA SULLO ZX80, ZX81 E ZX80-NUOVA ROM

Il programma che segue e' scritto per lo ZX80 e serve per gestire una agenda telefonica registrando nomi e numeri di telefono all'interno del programma. E' possibile creare ex novo l'agenda, aggiornarla, listarla. I nominativi non sono mantenuti a priori in ordine alfabetico, volendo possono essere ordinati. Fornendo il nome si ottiene il numero di telefono.

Alla fine del paragrafo sono segnalate le variazioni per far girare il programma sugli altri due calcolatori.

Nell'esempio si e' predisposto spazio per 200 indirizzi, volendo si puo' anche aumentare questo numero compatibilmente con le dimensioni della memoria. Il programma e' valido per il calcolatore fornito di espansione RAM di 16K. Per ogni nome e indirizzo sono necessari 40 byte, 28 per il nome e 12 per il telefono, ma a questi bisogna aggiungerne altri 6, 2 per il numero di linea della istruzione dove sta memorizzato il dato, 1 per la parola chiave PRINT, 2 per gli apici delimitatori del dato e 1 per il NEW LINE di chiusura. Quindi per ogni elemento dell'agenda servono 46 byte. Per 200 elementi servono 9200 byte (200*46=9200). Il programma inizia con una istruzione PRINT per mezzo della quale si puo' stampare il numero degli indirizzi presenti, tale numero sta nella variabile M, posta in fase di azzeramento al valore 0 e aggiornata in fase di inserimento o concellazione. Seguono poi 200 linee di PRINT seguite da 40 lineette tra apici; esse servono per memorizzare gli elementi dell'agenda. Vengono usati 28 caratteri per il nome e 12 caratteri per il telefono completando eventualmente con spazi le due zone.
L'utente puo' inizialmente usare un numero minore di 200 e scrivere meno PRINT per memorizzare. La linea 390 assegna ad N il valore 200; tale costante puo' essere modificata secondo le proprie esigenze.

Segue la codifica del programma:

```
   1 PRINT "SONO PRESENTI ";STR$(M);" NOMINATIVI"
   2 PRINT
   3 PRINT "40 lineette"
   4 PRINT "40 lineette"
   5 PRINT "40 lineette"
   ......
   ......
 201 PRINT "40 lineette"
 202 PRINT "40 lineette"
 203 GOSUB 280
```

```
204 GOTO 400
280 REM ROUTINE ATTESA PER VISIONE
283 PRINT "PREMI NEW LINE PER CONTINUARE"
285 INPUT A$
290 RETURN
300 REM ROUTINE RICERCA POSTO NOME
305 FOR K=1 TO N
306 LET L=(K-1)*46+P2
307 IF PEEK(L)=128 THEN GOTO 310
308 NEXT K
309 STOP
310 RETURN
320 REM ROUTINE RICERCA NOME UGUALE B$
321 LET I=P2
323 FOR K=1 TO N
324 LET B$=A$
325 FOR L=0 TO 27
326 IF CODE(B$)=1 THEN GOTO 350
327 IF PEEK(I+L)=CODE(B$) THEN GOTO 330
328 GOTO 335
330 LET B$=TL$(B$)
331 NEXT L
332 LET W=0
333 RETURN
335 LET I=I+46
340 NEXT K
345 LET W=1
347 RETURN
350 FOR L=L TO 27
353 IF PEEK(I+L)=0 THEN GOTO 360
355 GOTO 335
360 NEXT L
365 GOTO 332
370 REM ROUTINE SCRITTURA K1+1 CARATTERI
375 FOR K=0 TO K1
377 IF CODE (A$)=1 THEN GOTO 385
379 POKE L+K,CODE(A$)
380 LET A$=TL$(A$)
382 NEXT K
383 RETURN
385 FOR K=K TO K1
387 POKE L+K,0
388 NEXT K
389 GOTO 383
390 REM PROGRAMMA PRINCIPALE
392 LET N=200
395 REM P2 PUNTA A PRIMO ELEMENTO
398 LET P2=16474
400 CLS
405 PRINT "***AGENDA TELEFONICA***"
410 PRINT
420 PRINT "RISPONDI: 0 PER AZZERARE AGENDA"
```

```
 430 PRINT "            1 PER INSERIRE"
 440 PRINT "            2 PER CANCELLARE"
 450 PRINT "            3 PER RICERCARE"
 460 PRINT "            4 PER LISTARE"
 465 PRINT "            5 PER ORDINARE"
 467 PRINT "            9 PER USCIRE"
 470 INPUT A
 475 IF A = 9 THEN GOTO 5000
 480 IF A > 5 OR A < 0 THEN  GOTO 400
 490 GOTO (A+1)*500
 500 REM AZZERAMENTO
 503 CLS
 505 PRINT "AZZERAMENTO"
 507 PRINT "VUOI VERAMENTE AZZERARE ?"
 508 INPUT A$
 509 IF NOT A$="SI" THEN GOTO 400
 510 LET M=0
 515 LET I=P2
 520 FOR K= 1 TO N
 530 FOR L= I TO I+39
 540 POKE L,128
 550 NEXT L
 560 LET I= I+46
 570 NEXT K
 580 PRINT "TERMINATO AZZERAMENTO"
 585 GOSUB 280
 590 GOTO 400
1000 REM INSERIMENTO
1003 CLS
1005 PRINT "INSERIMENTO"
1010 PRINT "PER TERMINARE RISPONDI NEW LINE"
1025 IF M=N THEN GOTO 1110
1030 PRINT "NOME"
1040 INPUT A$
1050 IF A$="" THEN GOTO 400
1060 GOSUB 300
1065 LET M=M+1
1070 LET K1=27
1075 GOSUB 370
1077 LET L=L+28
1080 PRINT "TELEFONO"
1085 INPUT A$
1090 LET K1=11
1095 GOSUB 370
1100 GOTO 1003
1110 PRINT "FINITO POSTO"
1115 GOSUB 280
1125 GOTO 400
1500 REM CANCELLAZIONE
1503 CLS
1505 PRINT "PER TERMINARE RISPONDI NEW LINE"
1510 PRINT "NOME DA CANCELLARE"
```

```
1515 INPUT A$
1520 IF A$="" THEN GOTO 400
1530 LET W=0
1535 GOSUB 320
1540 IF NOT W=0 THEN GOTO 1570
1545 FOR K=0 TO 39
1550 POKE I+K,128
1555 NEXT K
1560 LET M=M-1
1565 GOTO 1503
1570 PRINT "NON TROVATO NOME ";A$
1572 GOSUB 280
1575 GOTO 1503
2000 REM RICERCA NOMI
2010 CLS
2015 PRINT "RICERCA NOMI"
2017 PRINT "PER USCIRE RISPONDI NEW LINE"
2020 PRINT "NOME DA RICERCARE"
2025 INPUT A$
2030 IF A$="" THEN GOTO 400
2040 LET W=0
2045 GOSUB 320
2050 IF NOT W=0 THEN GOTO 2150
2055 PRINT "TELEFONO: ";
2060 FOR K=0 TO 11
2065 LET L=PEEK(I+K+28)
2070 IF L=0 THEN GOTO 2085
2075 PRINT L-28;
2080 NEXT K
2085 PRINT
2090 GOTO 2010
2150 PRINT "NON TROVATO NOME ";A$
2153 GOSUB 280
2155 GOTO 2010
2500 REM LISTA RUBRICA
2510 PRINT "LISTA RUBRICA"
2520 GOTO 1
3000 REM ORDINAMENTO NOMI
3003 PRINT "ORDINAMENTO NOMI"
3004 PRINT "ATTENDI CON PAZIENZA"
3005 GOSUB 280
3007 LET K1=N-1
3010 LET W=0
3015 FOR K=1 TO K1
3020 LET L=(K-1)*46+P2
3025 FOR J=0 TO 27
3026 IF PEEK(L+J)<PEEK(L+J+46) THEN GOTO 3030
3027 IF PEEK(L+J)>PEEK(L+J+46) THEN GOTO 3100
3029 NEXT J
3030 NEXT K
3035 IF W=0 THEN GOTO 3200
3040 LET K1=K1-1
```

```
3050 IF K1=0 THEN GOTO 3200
3060 GOTO 3010
3100 LET W=1
3110 FOR J=0 TO 39
3115 LET A=PEEK(L+J)
3120 POKE L+J,PEEK(L+J+46)
3125 POKE L+J+46,A
3130 NEXT J
3135 GOTO 3030
3200 PRINT "FINITO ORDINAMENTO"
3205 GOSUB 280
3210 GOTO 400
5000 PRINT "FINE LAVORO"
5010 PRINT "PREPARA NASTRO PER REGISTRARE LA RUBRICA"
5015 PRINT "REGISTRA A VOCE IL NOME"
5020 PRINT "QUANDO SEI PRONTO PREMI NEW LINE"
5025 INPUT A$
5030 SAVE
```

Variabili usate nel programma:

. M numero nomi presenti nell'agenda, inizia a zero la prima volta che si usa il programma e viene mantenuto aggiornato.
. N numero indirizzi possibili, massimo 200; e' inizializzato alla linea 392.
. I variabile di comodo.
. A variabile di comodo
. A$ variabile per INPUT.
. B$ variabile stringa di comodo.
. K e J variabili controllo cicli.
. L variabile di controllo cicli e variabile di comodo.
. P2 puntatore al primo carattere del primo elemento agenda in linea 2 programma. Esso e' uguale a 16474, infatti il programma inizia in 16424, la linea 1 e' lunga 42 byte, la 2 e' lunga 4 byte e nella 3 prima del primo carattere dopo gli apici ci sono 4 byte (16424+42+4+4=16474).
. W variabile usata come flag, se 0 ha un significato se 1 un altro.
. K1 variabile di comodo.

Note al programma:

. Il programma deve essere fatto partire con GOTO 390. Appare il Menu' e cioe':

```
***AGENDA TELEFONICA***

Rispondi: 0 PER AZZERARE AGENDA
          1 PER INSERIRE
```

2 PER CANCELLARE
3 PER RICERCARE
4 PER LISTARE
5 PER ORDINARE
9 PER USCIRE

l'utente deve scegliere cosa vuole fare, ma la prima volta deve rispondere con 0 perche' la zona di memorizzazione dei dati deve essere riempita con spazi inversi, codice 128, altrimenti il programma non funziona bene. Qualora in seguito si scelga ancora l'opzione 0 si perde tutta l'agenda.

. La 1 serve per stampare il numero di elementi presenti nell'agenda; dopo STR$ tra parentesi si trova M, che e' la variabile dove sta memorizzato il numero degli elementi presenti.

. Le linee da 3 a 202 servono per tenere memorizzati i dati dell'agenda; essi sono inizialmente formati da 40 spazi. La routine di azzeramento pone in ogni PRINT tra gli apici 40 spazi inversi, codice 128. In tale modo in fase di ordinamento le linee non usate restano in fondo.

. La 203 rimanda al Menu' iniziale, infatti la parte di programma da 1 a 203 viene percorsa se si sceglie l'opzione 4 per listare.

. Da 280 a 290 si ha la routine di attesa NEW LINE che consente la visione dei messaggi; essa viene richiamata da diversi punti del programma. L'utente per proseguire deve premere NEW LINE.

. Da 300 a 310 si ha la routine per cercare una riga libera nell'agenda, riga libera significa iniziante con spazio inverso. All'uscita L punta alla posizione da usare per memorizzare un nuovo nome. Da questa routine si esce sicuramente con esito positivo, dato che non si entra se M=N.

. Da 320 a 365 si ha la routine per trovare nome uguale alla stringa B$. Si entra con W=0; se la ricerca e' stata positiva, si esce con W=0 e con I che punta alla posizione del primo carattere del nome. Se la ricerca e' stata negativa si esce con W=1.

. Da 370 a 389 si ha la routine che scrive in memoria K1+1 caratteri a partire dalla posizione L.

. A 390 inizia il programma principale e fino a 398 si ha l'inizializzazione di N e di P2. Ricordate che M viene inizializzato la prima volta dalla parte azzeramento.

. Da 400 a 490 si ha la presentazione del Menu' e la scelta della parte da eseguire in base alla risposta dell'utente. Le risposte possibili sono 0,1,2,3,4,5 e 9.

. RISPOSTA 0. Da 500 a 590 si ha il riempimento dell'agenda con spazi inversi e l'inizializzazione di M a zero.

. RISPOSTA 1. Da 1000 a 1125 si ha l'inserimento dei dati nell'agenda. Il dato viene inserito nel primo posto

libero trovato. Se non c'e' piu' posto viene segnalato. Vengono usate le routine a 300 e a 370.

. RISPOSTA 2. Da 1500 a 1575 si ha la cancellazione, si usa la routine a 320. Dove si cancella si mettono spazi inversi.

. RISPOSTA 3. Da 2000 a 2155 si ha la ricerca dei nomi e la stampa del telefono. Se non trova il nome lo segnala. Usa la routine a 320.

. RISPOSTA 4. Da 2500 a 2520 si lista la rubrica. Quando lo schermo e' pieno si deve premere 2 volte CONT.

. RISPOSTA 5. Da 3000 a 3210 si ha l'ordinamento dei nomi.

. RISPOSTA 9. Da 5000 a 5030 si ha il colloquio per memorizzare su nastro la nuova versione della rubrica insieme al programma. Se la rubrica e' stata usata solo per consultazione si esce con BREAK.

Per poter usare questo programma sullo ZX81 e ZX80-Nuova ROM si devono modificare:

. il valore del puntatore P2 nella linea 398 facendo i conti bene dato che le istruzioni sono piu' lunghe ed il programma inizia piu' avanti in memoria (P2=16562);

. il valore 46 da aggiungere per passare da una linea all'altra viene aumentato per le ragioni di cui sopra. Tale numero va modificato nelle linee 306, 335, 560, 3020, 3026, 3027, 3120, 3125;

. la routine che inizia a 320 dato che non si ha piu' disponibile la TL$ (326 IF CODE (B$)=11 THEN GOTO 350, 330 LET B$=B$(2 TO), 377 IF CODE(A$)=11 THEN GOTO 385, 380 LET A$=A$(2 TO))

. la routine che inizia a 370 per la stessa ragione;

. la linea 5015 sparisce e la linea 5030 diventa SAVE "RUBRICA".

Per contare quanti byte sono lunghe le istruzioni, nel nostro caso basta scrivere le seguenti istruzioni, per il nuovo Basic:

```
 1 PRINT "SONO PRESENTI ";STR$(M);" NOMINATIVI"
 2 PRINT
 3 PRINT "40 lineette"
50 FOR K = 16509 TO 16609
60 PRINT PEEK(K);" ";
70 NEXT K
```

sullo schermo appaiono i contenuti di 101 byte a partire da 16509, primo byte per il programma, si cerca il codice della prima lineetta dopo l'apice della linea 3 e si contano i byte fino alla lineetta; questo numero aggiunto a 16509 da' il valore del puntatore P2.

Ovviamente su questi calcolatori sarebbe meglio impostare

il programma usando, come suggerito nel paragrafo 9.14. le stringhe dimensionate per tenere memorizzati i dati. Puo' essere un utile esercizio realizzare un programma simile a questo, usando la tecnica delle stringhe con indice.

9.22. ANIMAZIONE DELLE FIGURE SULLO ZX80

Lo ZX80 e' organizzato in modo che quando lavora il calcolatore non vengono inviati fotogrammi allo schermo e quindi l'immagine scompare. Per poter ottenere il movimento delle figure si deve intervenire con una routine in linguaggio macchina che stabilizzi l'immagine sullo schermo in modo tale che si abbia l'impressione del movimento pur non essendo il quadro completamente persistente.

Nel programma che segue si fa rimbalzare una pallina chiara in un riquadro scuro disegnato sulle prime 22 linee dello schermo usando 31 colonne.

La codifica del programma e' la seguente:

```
  1 LET D=0
  2 LET U=0
  5 LET GG=33
  6 LET DX=1
  7 LET DY=1
 10 LET A=20270
 12 LET Y=1
 15 LET X=1
 20 LET S=A
 30 LET M$="CDE006CDC205012001D9CDC2051803CDAD01060810
           FE2A1E4023221E407CDE00C823DBFE3E3832234006
           5E10FED3FE3EEC06192A0C40CBFCCDAD013EF5042B
           FD352318CA"
 40 LET H=CODE(M$)
 50 IF H=1 THEN GOTO 200
 60 LET M$=TL$(M$)
 70 LET L=CODE(M$)
 80 POKE A,16*(H-28)+L-28
 90 LET M$=TL$(M$)
100 LET A=A+1
110 GOTO 40
200 CLS
205 LET C=16414
208 LET R=900
209 LET T=235
210 FOR K=0 TO 21
212 PRINT " ";
213 FOR J=1 TO 31
215 PRINT CHR$(128);
```

```
217 NEXT J
218 PRINT
219 NEXT K
220 LET I=0
225 LET D=PEEK(16397)*256+PEEK(16396)
230 LET D=D+2
250 POKE D+GG*Y+X,180
255 GOSUB R
260 POKE D+GG*Y+X,128
265 GOSUB R
275 LET X=X+DX
280 IF X=30 OR X=0 THEN LET DX=-DX
285 LET Y=Y+DY
290 IF Y=21 OR Y=0 THEN LET DY=-DY
300 GOTO 250
900 POKE C,T
910 POKE C+1,255
930 LET U=USR(S)
940 RETURN
```

La stringa M$ corrisponde al programma in linguaggio macchina che segue ed e' codificato in esadecimale. Nella codifica Assembler sono stati usati numeri esadecimali.

Indirizzo	Esadecimale	Assembler
20270	CD E0 06	CALL 06E0
20273	CD C2 05	CALL 05C2
20276	01 20 01	LD BC,0120
20279	D9	EXX
20280	CD C2 05	CALL 05C2
20283	18 03	JR 03
20285	CD AD 01	CALL 01AD
20288	06 08	LD B,08
20290	10 FE	DJNZ FE
20292	2A 1E 40	LD HL,(401E)
20295	23	INC HL
20296	22 1E 40	LD (401E),HL
20299	7C	LD A,H
20300	DE 00	SBC A,00
20302	C8	RET Z
20303	23	INC HL
20304	DB FE	IN A,(FE)
20306	3E 38	LD A,38
20308	32 23 40	LD 4023,A
20311	06 5E	LD B,5E
20313	10 FE	DJNZ FE
20315	D3 FE	OUT (FE),A
20317	3E EC	LD A,EC
20319	06 19	LD B,19
20321	2A 0C 40	LD HL,(400C)

```
20324        CB FC         SET 7,H
20326        CD AD 01      CALL 01AD
20329        3E F5         LD A,F5
20331        04            INC B
20332        2B            DEC HL
20333        FD 35 23      DEC (IY+23)
20336        18 CA         JR CA
```

Variabili usate nel programma:

. D indirizzo iniziale DISPLAY-FILE.
. U variabile di comodo per la funzione USR.
. GG avanzamento Y.
. DX incremento per X.
. DY incremento per Y.
. X e Y coordinate posizione sul video: X si riferisce alle colonne e Y alle righe.
. A locazione di partenza (20270) e puntatore per memorizzare il programma in linguaggio macchina contenuto in M$.
. S indirizzo per la funzione USR.
. M$ stringa contenente la codifica esadecimale del programma in codice macchina.
. H e L variabili di comodo.
. C indirizzo contatore fotogrammi dello schermo. Il contatore dei fotogrammi viene usato come timer.
. R indirizzo del sottoprogramma che crea una pausa.
. T tempo per la pausa, viene usato anche nel programma in linguaggio macchina prelevando il valore del contatore dei fotogrammi.

Note al programma:

. Le linee 10 e da 20 a 110 caricano il programma in linguaggio macchina scritto in codice esadecimale. Esse possono essere utilizzate in qualunque programma per lo stesso scopo; basta modificare A per definire l'indirizzo di inizio della memorizzazione del codice macchina, e il contenuto di M$.
. Le variabili usate nel programma sono tutte predefinite prima di usare la routine che disegna sul video, per evitare di spostare il display file. Da 210 a 219 viene disegnato un rettangolo nero (spazi inversi) di 22x31 posizioni.
. Da 250 a 300 viene disegnata la pallina (una O in campo inverso), cancellandola dalla precedente posizione. Il tempo di permanenza dell'immagine dipende da T.
. Da 900 a 910 viene preparato il contatore dei fotogrammi dello schermo in modo che contenga un numero negativo che dipende dal valore di T.
. La linea 930 chiama la routine in linguaggio macchina.

. Il byte 16420 contiene la coordinata X della colonna della posizione attuale sul video partendo dal valore 33 a sinistra e arrivando a 2 a destra. Il byte 16421 contiene la coordinata Y della riga della posizione attuale sul video partendo dal valore 23 in alto ed arrivando al valore 0 in basso.

. A 20270 si ha la chiamata alla routine che fornisce la posizione corrente sul video.

. A 20273 si ha la chiamata alla routine di completamento del display file.

. Da 20276 a 20280 sistema i registri B' e C' e chiama ancora la routine di completamento del display file.

. Da 20283 a 20290 invia un fotogramma al video e crea una pausa.

. Da 20292 a 20296 incrementa il contatore dei fotogrammi.

. Da 20299 a 20302 se il contatore si e' azzerato ritorna al Basic.

. Da 20303 a 20315 manda segnali al sistema e crea una attesa per sincronizzare il video.

. Da 20317 a 20326 prepara HL e manda un fotogramma al video.

. Da 20329 a 20336 prepara B e HL e poi torna 20285.

In realta' con questo programma non si riesce ad ottenere una buona stabilita' dello schermo; si puo' provare a modificare T per vedere se la situazione migliora.

Per ottenere una buona stabilita' si dovrebbe programmare completamente in linguaggio macchina senza tornare mai al Basic.

9.23. RINUMERAZIONE LINEE PROGRAMMA BASIC SULLO ZX80

Il problema della rinumerazione delle linee di un programma Basic sarebbe semplice se non ci fossero le istruzioni GOTO e GOSUB. Un programma di rinumerazione ben fatto deve sistemare anche i richiami ai numeri di linea presenti nelle istruzioni. Una complicazione deriva dal fatto che, mentre il numero di linea, di inizio linea programma, e' memorizzato in due byte con la parte intera del numero di linea diviso 256 nel primo byte e con il resto della precedente divisione nel secondo byte (in tale modo tutti i numeri da 1 a 9999 occupano lo stesso spazio), i numeri di linea dopo i GOTO/GOSUB sono memorizzati carattere per carattere. Per quest'ultima ragione per passare da numero linea 10 a numero linea 150 occorre un byte in piu'. Questo byte in piu' (e potrebbe essere anche in meno) si puo' trovare spostando il la' (o in qua') tutta la parte restante del programma. Infatti, mentre lavorando sotto

sistema le modifiche al programma comportano automaticamente (cioe' senza che l'utente se ne accorga) lo spostamento delle linee di programma in memoria per guadagnare o perdere spazio, lavorando sotto programma questo non avviene. Bisogna tenere presente che quando il programma si sposta in memoria si spostano anche le altre aree, zona variabili, memoria di schermo,ecc., e quindi cambiano gli indirizzi contenuti nei puntatori.

L'argomento della rinumerazione delle linee di un programma puo' servire di spunto per riflettere sul modo nel quale lavora il sistema Basic e quindi vale la pena di occuparsene.

Nel primo esempio che segue viene rinumerato un programma segnalando al video dove e come correggere i richiami interni manualmente. Nel secondo esempio viene suggerito come ottenere da programma anche questo lavoro.

PRIMO ESEMPIO.

Per poter eseguire correttamente il lavoro si deve scandire il programma e preparare una tabella dei GOTO e GOSUB presenti, memorizzando:
. il numero di linea del GOTO o GOSUB;
. il numero di linea a cui manda il GOTO o il GOSUB;
. lasciando libero lo spazio per poter memorizzare il nuovo numero di linea da sostituire al vecchio;
. lasciando libero lo spazio per poter memorizzare la nuova destinazione dopo il GOTO/GOSUB;
occorre quindi una tabellina di 4 elementi per ogni GOTO/GOSUB da sistemare.
Durante la prima scansione, che ha luogo solo se N e' diverso da zero, il programma Basic non viene modificato in memoria. La tabellina dei GOTO/GOSUB deve avere delle dimensioni che sono richieste all'inizio della routine di rinumerazione. Con questo numero, N, viene dimensionato un vettore con 4*N+4 elementi.
Dopo si ha il ciclo di rinumerazione di 10 in 10 e, per ogni linea rinumerata, se il numero N e' diverso da zero, si va a completare la tabellina nelle due caselle lasciate libere. Alla fine della rinumerazione viene evidenziata sul video la tabellina per poter apportare manualmente le modifiche necessarie.
Ricordiamo che il programma inizia al byte 16424, che ogni istruzione inizia con due byte contenenti il numero di linea, che ogni istruzione termina con 118, che i numeri di linea dopo il GOTO o il GOSUB sono memorizzati carattere per carattere, che il codice del GOTO e' 236 e che il codice del GOSUB e' 251.

Il programma di rinumerazione viene scritto partendo dal numero di linea 9000 e viene mandato in esecuzione con RUN 9000. Il programma da rinumerare deve avere numeri di linea minori di 9000.

Codifica del programma:

```
9000 REM RINUMERAZIONE
9005 CLR
9010 LET T1=16424
9015 PRINT "QUANTI GOTO/GOSUB"
9020 INPUT N
9025 IF N=0 THEN GOTO 9135
9030 DIM T(4*N+4)
9035 LET J=1
9040 LET K=T1
9045 LET T2=PEEK(K)*256+PEEK(K+1)
9050 IF T2=9000 THEN GOTO 9135
9055 LET K=K+2
9060 IF PEEK(K)=236 OR PEEK(K)=251 THEN GOTO 9080
9065 IF PEEK(K)=118 THEN GOTO 9110
9070 LET K=K+1
9075 GOTO 9060
9080 LET T(J)=T2
9085 LET K=K+1
9090 LET T3=PEEK(K)
9095 IF T3<28 OR T3>37 THEN GOTO 9120
9100 LET T(J+1)=T(J+1)*10+T3-28
9105 GOTO 9085
9110 LET K=K+1
9115 GOTO 9045
9120 LET J=J+4
9125 IF J>4*N THEN GOTO 9135
9130 GOTO 9065
9135 LET T4=0
9140 LET K=T1
9145 LET T4=T4+10
9150 LET T2=PEEK(K)*256+PEEK(K+1)
9155 IF T2=9000 THEN GOTO 9235
9160 POKE K,T4/256
9165 POKE K+1, T4-(T4/256)*256
9170 IF N=0 THEN GOTO 9205
9175 LET J=1
9180 IF T(J+1)=T2 THEN LET T(J+3)=T4
9185 IF T(J)=T2 THEN LET T(J+2)=T4
9190 LET J=J+4
9195 IF J>N*4 THEN GOTO 9205
9200 GOTO 9180
9205 LET K=K+2
9210 IF PEEK(K)=118 THEN GOTO 9225
9215 LET K=K+1
9220 GOTO 9210
```

```
9225 LET K=K+1
9230 GOTO 9145
9235 IF N=0 THEN GOTO 9275
9240 PRINT "MODIFICARE LE SEGUENTI LINEE"
9245 PRINT "VECCHIA","DES."," NUOVA","DEST."
9250 LET J=1
9255 PRINT T(J),T(J+1),T(J+2),T(J+3)
9260 LETJ=J+4
9265 IF J > 4*N THEN STOP
9270 GOTO 9255
9275 PRINT "FINE"
```

Variabili usate nel programma:

- N per il numero dei GOTO e GOSUB da modificare;
- T(N*4+4) tabella GOTO/GOSUB;
- T(1) numero vecchio linea;
- T(2) numero vecchio destinazione GOTO/GOSUB;
- T(3) numero nuovo della linea;
- T(4) numero nuovo della destinazione GOTO/GOSUB;
- J variabile controllo ciclo;
- K variabile controllo ciclo;
- T1 indirizzo inizio programma (16424);
- T2 numero vecchio di linea;
- T3 cifra del numero linea dopo GOTO/GOSUB;
- T4 numero di linea dopo GOTO/GOSUB;

Note al programma:

. Da 9000 a 9020 vengono azzerate le variabili, inizializzato T1 e viene chiesto quanti GOTO e GOSUB ci sono nel programma, tale numero si trova in N;

. La linea 9025 fa saltare la preparazione della tabellina dei GOTO/GOSUB se N=0

. Da 9030 a 9130 viene scandito il programma e viene riempita la tabella dei GOTO/GOSUB nella prima e seconda posizione;

. Da 9135 a 9230 viene rinumerato il programma e se N diverso da zero completata la tabellina dei GOTO/GOSUB nella terza e quarta posizione;

. La linea 9235 fa saltare la stampa della tabella se N=0 e manda al messaggio finale in 9275

. Da 9240 a 9270 viene stampata la tabellina delle modifiche da fare manualmente.

Se si desidera memorizzare su nastro il programma modificato, si deve cancellare il programma di rinumerazione da 9000 a 9275.

SECONDO ESEMPIO

In questo caso il programma si compone delle seguenti parti:

. a) scansione del programma per riempire la tabellina dei GOTO/GOSUB, sempre che l'utente dica che essi sono presenti, la tabellina di memorizzazione deve contenere:
- il numero vecchio della linea del GOTO/GOSUB,
- il numero vecchio di linea dopo il GOTO/GOSUB,
- il numero nuovo della linea del GOTO/GOSUB,
- la nuova destinazione del GOTO/GOSUB;

. b) rinumerazione del programma con completamento tabellina GOTO/GOSUB, se N diverso da zero;

. c) se N diverso da zero, sistemazione delle destinazioni dei GOTO/GOSUB spostando la parte restante del programma in giu' o in su' a seconda dei casi. Solo che quest'ultimo lavoro presenta una certa complessita'. Infatti se noi ci mettiamo a spostare il programma che precede la linea 9000 in memoria andiamo ad invadere la linea 9000 se il programma si allunga e creiamo dei byte senza senso se esso si accorcia. Per ovviare all'inconveniente si dovrebbe fare iniziare il programma di rinumerazione con una linea 9000 formata da 9000 REM e poi, per esempio, 50 P. Tale linea serve come polmone per ricuperare dei byte. All'inizio della sistemazione dei GOTO/GOSUB si dovrebbe calcolare in base alla tabellina che reca i vecchi numeri di linea di destinazione ed i nuovi di quanti byte in giu' o in su e' lo spostamento. Se lo spostamento e' in giu' si deve spostare 9000 e REM di quei byte, diminuendo cosi' i 50 P, ma lasciando una istruzione Basic valida. Poi si puo' tranquillamente procedere allo spostamento del programma in base alla sistemazione dei numeri di linea. Alla fine tutto e' a posto. Se invece il programma si accorcia, prima si deve procedere alla sistemazione del programma e poi andare a spostare 9000 e REM all'indietro aggiungendo dei P. Naturalmente il numero dei P usati nella prima istruzione deve essere sufficiente a coprire le necessita'.

In realta' il contenuto di questo paragrafo non presenta una grande utilita' pratica per sistemi come i nostri Sinclair dove non esiste ancora la possibilita' di fondere file di programmi. Stando cosi' le cose un programma di utilita' come quello della rinumerazione diventa un po' pesante da riscrivere ogni volta che serve, per aggiungerlo al proprio programma da rinumerare. L'interesse del paragrafo sta invece nell'essersi soffermati su argomenti che lasciano intravedere come lavora l'interprete Basic.

Questi esempi non sono molto semplici, ma puo' essere un buon esercizio di programmazione capire come funzionano.
Con le necessarie modifiche questi programmi possono essere usati anche per lo ZX81 e ZX80-Nuova ROM.
Nel Capitolo 8 e' riportato un esempio di rinumerazione per il nuovo Basic in linguaggio macchina.

9.24. USO DELLA FUNZIONE INKEY$

Questa funzione prende il carattere disponibile alla tastiera quando il programma la esegue. Esempio:

```
10 IF INKEY$ = "" THEN GOTO 10
20 PRINT "NO STRINGA NULLA"
```

se date il RUN a questo piccolo programma, vedete subito sul video:

NO STRINGA NULLA

infatti quando viene eseguita la 10 sulla tastiera permane il NEW LINE che voi avete premuto dopo RUN. Il programma parte prima che voi togliete il dito dalla tastiera e quindi viene sentito un NEW LINE che non e' la stringa nulla ed il programma prosegue.

Provate a scrivere:

```
10 IF INKEY$ <> "" THEN GOTO 10
20 IF INKEY$ = "" THEN GOTO 20
30 PRINT "NO STRINGA NULLA"
```

se date il RUN non vedete caratteri sullo schermo; se premete un qualunque tasto vedrete la scritta NO STRINGA NULLA. Infatti la linea 10 blocca l'effetto temporale del tasto NEW LINE e la linea 20 crea l'attesa fino a quando premete un qualunque tasto.

Potete fare la seguente prova:

```
10 DIM A$(50)
20 FOR K=1 TO 50
30 LET A$(K)=INKEY$
40 NEXT K
50 FOR K=1 TO 50
60 PRINT CODE A$(K);" ";
70 NEXT K
75 PRINT
80 STOP
```

date il RUN al programma e subito dopo il NEW LINE premete a caso qualche tasto cercando di essere veloci. I vostri tasti vengono memorizzati nella tabella A$ e il ciclo di stampa ve ne mostra il codice. All'inizio vedrete un certo numero di 118; il loro numero dipende dal tempo di permanenza del vostro dito sul tasto del NEW LINE. Probabilmente dopo vedrete qualche zero, codice della stringa nulla, e poi tra altri gruppi di zeri il codice ripetuto dei tasti premuti.

Provate per esempio:

```
10 FOR K=1 TO 10
20 NEXT K
30 IF INKEY$ = "" THEN 30
40 PRINT "NO STRINGA NULLA".
```

vedrete che funziona cioe' non appare la scritta fino a quando non premete un tasto. Infatti il ciclo 10/20 ha esaurito l'effetto temporale del NEW LINE usato dopo RUN. Se diminuite il numero limite nel FOR e passate da 10 a 3 non funziona; evidentemente 3 non basta.

APPENDICE A

C A R A T T E R I
D E L   S I S T E M A

Riportiamo "TABELLA 1" dei caratteri del sistema. Per ogni codice sono elencati:

. nelle 2 colonne ZX80 e ZX81 il carattere corrispondente nei due calcolatori o un riferimento alle note;

. il corrispondente codice esadecimale.

Nella colonna "caratteri o note" si rimanda alle note con *n). In questa stessa colonna sono elencate anche le parole chiave e le funzioni del linguaggio BASIC, infatti anche questi elemementi sono codificati con un carattere ASCII.

Nella Tabella 1 si usano le seguenti abbreviazioni:
crs. sta per cursore;
inv. sta per campo inverso.

Si ricorda che ogni carattere e' memorizzato in un byte (8 bit) e che un byte puo' contenere un numero decimale compreso tra 0 e 255 (e quindi un numero esadecimale compreso tra 0 e FF).

Come si vede nella Tabella 1 non tutte le configurazioni di bit corrispondono a caratteri stampabili.

Nella Appendice F e' riportata la codifica del linguaggio macchina, e, ovviamente, anche in questa i codici vanno da 0 a 255 (da 0 a FF in esadecimale). Quando il calcolatore lavora in BASIC l'interpretazione dei codici e' quella riportata in Tabella 1, mentre quando il calcolatore lavora in linguaggio macchina l'interpretazione e' quella riportata nella Appendice F.

Con il programma:

```
10 INPUT X
15 IF X=0 THEN GOTO 30
20 PRINT CHR$(X)
25 GOTO 10
30 STOP
```

si puo' stampare il carattere o la parola corrispondente al codice X. Per i caratteri contrassegnati da *3) si ottiene il simbolo ? (punto interrogativo).

Con il programma:

```
10 INPUT X
15 INPUT Y
20 FOR K = X TO Y
25 PRINT CHR$(K),
30 NEXT K
40 STOP
```

si possono stampare i caratteri che hanno il codice compreso tra X e Y.

Significato delle note:

. 1) Sono disponibili 22 caratteri grafici: lo spazio (CHR$(0)) e lo spazio inverso (quadratino nero, CHR$(128)) hanno la stessa codifica nei due calcolatori; gli altri caratteri grafici no. Segue la "TABELLA 2" dei caratteri grafici e la loro codifica.

. 2) Nei due calcolatori CHR$(12) rappresenta il carattere lira (L maiuscola tagliata) e CHR$(140) lo stesso carattere in campo inverso.

. 3) Sono configurazioni di carattere non usate, se si tenta di stamparli con CHR$(X) si ottiene il punto interrogativo.

. 4) Questo carattere "", rappresenta gli apici da introdurre nelle stringhe. Esso non va usato come stringa nulla, anche se si e' tentati di farlo. La stringa nulla si ottiene battendo 2 volte gli apici.

Seguono le due tabelle.

TABELLA 1

Cod. dec.	Carattere o note ZX80	Carattere o note ZX81	Cod. esad.	Cod. dec.	Carattere o note ZX80	Carattere o note ZX81	Cod. esad.
0	spazio	spazio	00	38	A	A	26
1	"	*1)	01	39	B	B	27
2	*1)	*1)	02	40	C	C	28
3	*1)	*1)	03	41	D	D	29
4	*1)	*1)	04	42	E	E	2A
5	*1)	*1)	05	43	F	F	2B
6	*1)	*1)	06	44	G	G	2C
7	*1)	*1)	07	45	H	H	2D
8	*1)	*1)	08	46	I	I	2E
9	*1)	*1)	09	47	J	J	2F
10	*1)	*1)	0A	48	K	K	30
11	*1)	"	0B	49	L	L	31
12	*2)	*2)	0C	50	M	M	32
13	$	$	0D	51	N	N	33
14	:	:	0E	52	O	O	34
15	?	?	0F	53	P	P	35
16	(	(	10	54	Q	Q	36
17	)	)	11	55	R	R	37
18	>	>	12	56	S	S	38
19	<	<	13	57	T	T	39
20	=	=	14	58	U	U	3A
21	+	+	15	59	V	V	3B
22	-	-	16	60	W	W	3C
23	*	*	17	61	X	X	3D
24	/	/	18	62	Y	Y	3E
25	;	;	19	63	Z	Z	3F
26	,	,	1A	64	*3)	RND	40
27	.	.	1B	65	*3)	INKEY$	41
28	0	0	1C	66	*3)	PI	42
29	1	1	1D	67	*3)	*3)	43
30	2	2	1E	68	*3)	*3)	44
31	3	3	1F	69	*3)	*3)	45
32	4	4	20	70	*3)	*3)	46
33	5	5	21	71	*3)	*3)	47
34	6	6	22	72	*3)	*3)	48
35	7	7	23	73	*3)	*3)	49
36	8	8	24	74	*3)	*3)	4A
37	9	9	25	75	*3)	*3)	4B

Cod. dec.	Carattere o note ZX80	ZX81	Cod. esad.	Cod. dec.	Carattere o note ZX80	ZX81	Cod. esad.
76	*3)	*3)	4C	121	*3)	FUNCTION	79
77	*3)	*3)	4D	122	*3)	*3)	7A
78	*3)	*3)	4E	123	*3)	*3)	7B
79	*3)	*3)	4F	124	*3)	*3)	7C
80	*3)	*3)	50	125	*3)	*3)	7D
81	*3)	*3)	51	126	*3)	*3)	7E
82	*3)	*3)	52	127	*3)	*3)	7F
83	*3)	*3)	53	128	sp. inv.	*1)	80
84	*3)	*3)	54	129	" inv.	*1)	81
85	*3)	*3)	55	130	*1)	*1)	82
86	*3)	*3)	56	131	*1)	*1)	83
87	*3)	*3)	57	132	*1)	*1)	84
88	*3)	*3)	58	133	*1)	*1)	85
89	*3)	*3)	59	134	*1)	*1)	86
90	*3)	*3)	5A	135	*1)	*1)	87
91	*3)	*3)	5B	136	*1)	*1)	88
92	*3)	*3)	5C	137	*1)	*1)	89
93	*3)	*3)	5D	138	*1)	*1)	8A
94	*3)	*3)	5E	139	*1)	" inv.	8B
95	*3)	*3)	5F	140	*2)	*2)	8C
96	*3)	*3)	60	141	$ inv.	$ inv.	8D
97	*3)	*3)	61	142	: inv.	: inv.	8E
98	*3)	*3)	62	143	? inv.	? inv.	8F
99	*3)	*3)	63	144	(inv.	(inv.	90
100	*3)	*3)	64	145	) inv.	) inv.	91
101	*3)	*3)	65	146	> inv.	> inv	92
102	*3)	*3)	66	147	< inv.	< inv.	93
103	*3)	*3)	67	148	= inv.	= inv.	94
104	*3)	*3)	68	149	+ inv.	+ inv.	95
105	*3)	*3)	69	150	- inv.	- inv.	96
106	*3)	*3)	6A	151	* inv.	* inv.	97
107	*3)	*3)	6B	152	/ inv.	/ inv.	98
108	*3)	*3)	6C	153	; inv.	; inv.	99
109	*3)	*3)	6D	154	, inv.	, inv	9A
110	*3)	*3)	6E	155	. inv.	. inv.	9B
111	*3)	*3)	6F	156	0 inv.	0 inv	9C
112	*3)	crs.su'	70	157	1 inv.	1 inv.	9D
113	*3)	crs.giu'	71	158	2 inv.	2 inv.	9E
114	*3)	crs.sin.	72	159	3 inv.	3 inv.	9F
115	*3)	crs.dest.	73	160	4 inv.	4 inv.	A0
116	*3)	GRAPHICS	74	161	5 inv.	5 inv.	A1
117	*3)	EDIT	75	162	6 inv.	6 inv.	A2
118	*3)	NEWLINE	76	163	7 inv.	7 inv.	A3
119	*3)	RUBOUT	77	164	8 inv.	8 inv.	A4
120	*3)	stato K/L	78	165	9 inv.	9 inv.	A5

Cod. dec.	Carattere o note		Cod. esad.	Cod. dec.	Carattere o note		Cod. esad.
	ZX80	ZX81			ZX80	ZX81	
166	A inv.	A inv.	A6	211	*3)	PEEK	D3
167	B inv.	B inv.	A7	212	"	USR	D4
168	C inv.	C inv.	A8	213	THEN	STR$	D5
169	D inv.	D inv.	A9	214	TO	CHR$	D6
170	E inv.	E inv.	AA	215	;	NOT	D7
171	F inv.	F inv.	AB	216	,	**	D8
172	G inv.	G inv.	AC	217	)	OR	D9
173	H inv.	H inv.	AD	218	(	AND	DA
174	I inv.	I inv.	AE	219	NOT	<=	DB
175	J inv.	J inv.	AF	220	-	>=	DC
176	K inv.	K inv.	B0	221	+	<>	DD
177	L inv.	L inv.	B1	222	*	THEN	DE
178	M inv.	M inv.	B2	223	/	TO	DF
179	N inv.	N inv.	B3	224	AND	STEP	E0
180	O inv.	O inv.	B4	225	OR	LPRINT	E1
181	P inv.	P inv.	B5	226	**	LLIST	E2
182	Q inv.	Q inv.	B6	227	=	STOP	E3
183	R inv.	R inv.	B7	228	>	*3)	E4
184	S inv.	S inv.	B8	229	<	*3)	E5
185	T inv.	T inv.	B9	230	LIST	NEW	E6
186	U inv.	U inv.	BA	231	RETURN	SCROLL	E7
187	V inv.	V inv.	BB	232	CLS	CONT	E8
188	W inv.	W inv.	BC	233	DIM	DIM	E9
189	X inv.	X inv.	BD	234	SAVE	REM	EA
190	Y inv.	Y inv.	BE	235	FOR	FOR	EB
191	Z inv.	Z inv.	BF	236	GOTO	GOTO	EC
192	*3)	*4)	C0	237	POKE	GOSUB	ED
193	*3)	AT	C1	238	INPUT	INPUT	EE
194	*3)	TAB	C2	239	RANDOMISE	LOAD	EF
195	*3)	*3)	C3	240	LET	LIST	F0
196	*3)	CODE	C4	241	*3)	LET	F1
197	*3)	VAL	C5	242	*3)	PAUSE	F2
198	*3)	LEN	C6	243	NEXT	NEXT	F3
199	*3)	SIN	C7	244	PRINT	POKE	F4
200	*3)	COS	C8	245	*3)	PRINT	F5
201	*3)	TAN	C9	246	NEW	PLOT	F6
202	*3)	ASN	CA	247	RUN	RUN	F7
203	*3)	ACS	CB	248	STOP	SAVE	F8
204	*3)	ATN	CC	249	CONTINUE	RAND	F9
205	*3)	LN	CD	250	IF	IF	FA
206	*3)	EXP	CE	251	GOSUB	CLS	FB
207	*3)	INT	CF	252	LOAD	UNPLOT	FC
208	*3)	SQR	D0	253	CLEAR	CLEAR	FD
209	*3)	SGN	D1	254	REM	RETURN	FE
210	*3)	ABS	D2	255	*3)	COPY	FF

TABELLA 2

Codice		Simbolo	Codice		Simbolo
ZX80	ZX81		ZX80	ZX81	
2	5		3	131	
4	1		5	2	
6	4		7	7	
8	6		9	8	
10	9		11	10	
130	133		131	3	

200

Nello ZX80 i caratteri grafici si ottengono premendo i relativi tasti insieme allo SHIFT. Nello ZX81 e ZX890-Nuova ROM i caratteri grafici si ottengono dopo aver posto il cursore nello stato G (con SHIFT e GRAPHICS) usando i relativi tasti insieme allo SHIFT.

Per espandere i codici dei caratteri delle parole chiave del Basic il sistema si serve di una tabella memorizzata in ROM.

APPENDICE B

V A R I A B I L I
D E L S I S T E M A

VARIABILI DEL SISTEMA PER LO ZX80

Il contenuto dei primi 40 byte della RAM e' quello spiegato qui di seguito. Alcune variabili occupano un byte, altre due. Con le istruzioni POKE e PEEK si possono scrivere e leggere queste variabili. Se le variabili sono di 1 byte non ci sono problemi. Se le variabili sono di due byte per scrivere una variabile di valore V all'indirizzo n si deve procedere cosi':

```
POKE n+1,INT(V/256)         si scrive la parte intera di V/256
POKE n,V-256*INT(V/256)     si scrive il resto della divisione
                            precedente
```

Analogamente per conoscere il valore V di una variabile di 2 byte di indirizzo n, si deve procedere cosi: PEEK(n) + PEEK(n+1)*256, se si e' sicuri che la variabile e' positiva. Se la variabile puo' essere negativa si deve invece procedere cosi':

```
LET MSB = PEEK(n+1)
IF MSB > 127 THEN LET MSB = MSB - 256
LET V = PEEK(n) + MSB * 256.
```

Nella tabella che segue valgono queste convenzioni:

X   significa che la variabile non puo'essere modificata, se lo si fa il sistema si blocca;

N   si puo' anche modificare la variabile dato che il sistema la rigenera;

1 o 2 a indicare se occupa 1 o 2 byte;

U   a indicare variabile non segnata, cioe' da 0 a 65535, il BASIC tratta questa variabile considerando i va= lori da 32768 a 65535 come valori da -32768 a -1.

Note	Indirizzi	Commenti
1	16384	contiene il numero dell'errore accaduto -1, normalmente contiene 255. Se capita un er= rore di supero di capacita', codice 6, essa contiene 5. Se si ha una POKE per scriverci qualcosa si devono usare solo i numeri 255 oppure tra 0 e 8. Se si scrive POKE16384,8 si ha STOP,infatti 8+1=9 codice dello STOP.
X1	16385	flag usati dal sistema, cioe' indicatori interni.
2	16386	numero di linea in esecuzione. POKE non ha effetto a meno che non sia nell'ultima li= nea del programma.
N2	16388	posizione in RAM (zona video) del cursore K o L dello schermo.
2	16390	numero della linea alla quale si trova il puntatore di linea.
X2	16392	VARS si veda Cap. 7.
X2	16394	E-LINE si veda Cap. 7.
X2	16396	D-FILE si veda Cap. 7.
X2	16398	DF-EA ssi veda il Cap. 7.
X2	16400	DF-END si veda Cap. 7.
X1	16402	numero di linee della parte bassa dello schermo, inclusa la linea in bianco che se= para le due parti.
2	16403	numero della linea che appare per prima sullo schermo. Viene modificato da LIST e quando lo schermo elimina le prime linee.
2	16405	indirizzo di quello che precede il cursore marcatore di errore S.
2	16407	numero della linea alla quale fa saltare CONTINUE.
N1	16409	flag usati dal sistema per controllare la sintassi delle frasi.
N2	16410	indirizzo del prossimo elemento nella ta= bella della sintassi.

Note	Indirizzi	Commenti
U2	16412	punto di partenza per il generatore dei nu= meri random. Viene modificato da RANDOMISE ed aggiornato ogni volta che si usa RND.
U2	16414	numero dei fotogrammi dello schermo dal mo= mento dell' accensione dello ZX80. Piu' esattamente il resto quando questo e' divi= so per 65535. Quando si ha una immagine sullo schermo, questo contatore e' incre= mentato 50 volte al secondo nella versione UK e 60 volte al secondo nella versione US.
N2	16416	indirizzo del primo carattere del nome del= la prima variabile in frasi LET,INPUT,FOR, NEXT,DIM.
N2	16418	valore dell'ultima espressione o variabile.
X1	16420	posizione nella linea attuale del prossimo carattere da scrivere sullo schermo, dove 33 significa ultima a sinistra, 32 la se= conda da sinistra, 2 l'ultima a destra, 1 prima colonna della prossima linea perche' la linea attuale e' piena, 0 prima colonna nella prossima linea perche' e' arrivato il segnale di fine linea (dopo una PRINT che non termina con virgola o con punto e vir= gola). Si ha 33 se lo schermo e' vuoto, per esempio dopo un CLS.
X1	16421	posizione della linea attuale sullo schermo 23 significa linea in alto, 22 seconda li= nea, ecc.
X2	16422	indirizzo del carattere dopo la parentesi chiusa in PEEK oppure del NEW LINE alla fi= ne della frase POKE.

NOTA: Si segnala l'algoritmo usato per generare i numeri pseudo-random. Sia n il valore contenuto in 16412; se n=0 si pone n=65536.
Si calcola il resto m di (n*77) / 65537 , se m=65536 si pone m=0. Il risultato di RND(x) e' X * m/65536. m viene posto in 16412.

VARIABILI DEL SISTEMA PER LO ZX81

La memoria RAM del sistema inizia con l'indirizzo 16384. I primi 125 byte della RAM sono usati dal sistema per scopi particolari, la zona utente inizia all'indirizzo 16509.
Nella tabella che segue sono descritte le "Variabili del Sistema", alcune di esse occupano 1 byte, altre 2 byte. Se la variabile occupa 2 byte essa e' indirizzata dal byte con indirizzo minore (contrariamente a quanto si sarebbe portati a pensare) e questo e' il meno significativo. Per mezzo delle istruzioni PEEK e POKE si possono leggere e scrivere (se e' consentito) queste variabili.
Si ricorda che, se la variabile occupa 2 byte, per scrivere un valore V in essa si deve procedere cosi':

```
POKE n+1,INT(V/256)       si scrive la parte intera di V/256
POKE n,V-256*INT(V/256)   si scrive il resto della divisione
                          precedente
```

Analogamente per ottenere il valore V di una variabile occupante due byte (di indirizzo n e n+1) si deve procedere cosi':

```
PEEK n + 256*PEEK(n+1)
```

se si e' sicuri che la variabile e' positiva; altrimenti per ottenere un valore V corretto si deve procedere cosi':

```
LET MSB=PEEK(n+1)
IF MSB > 127 THEN LET MSB = MSB -256
LET V = PEEK n + MSB*256
```

Nella tabella viene dato un nome mnemonico ad ogni variabile del sistema solo per comodita' di riferimento, tali nomi ovviamente non possono essere usati nei programmi BASIC. Le variabili del sistema sono accessibili solo tramite i comandi POKE e PEEK.
Nella colonna "Note" della tabella possono comparire delle lettere maiuscole aventi il seguente significato:

- X la variabile non puo' essere modificata;
- N la variabile puo' essere modificata senza danno;
- S la variabile viene conservata quando si usa il comando SAVE.

Inoltre, nella stessa colonna, compare un numero che indica quanti byte sono usati per la variabile o la zona di memoria identificata.

Note	Indirizzo	Nome	Contenuto
1	16384	ERR-NR	Numero del codice di errore - 1. Di norma contiene 255. Con la frase POKE 16384,n si puo' forzare uno STOP. Se 0<=n<=14 si ottiene uno dei messaggi standard. Se 15<=n<-34 o 99<=n<-127 si hanno messaggi non standard. Se 35<=n<=98 si ottiene un collega=mento alla memoria di schermo.
X 1	16385	FLAGS	Indicatori usati dal BASIC.
X 2	16386	ERR-SP	Indirizzo del primo argomento nella STACK area, dopo i GOSUB/RETURN.
2	16388	RAMTOP	Indirizzo del primo byte sopra la zona del BASIC. Se si fa una POKE in RAMTOP essa ha effetto al primo comando NEW o CLS.
N 1	16390	MODE	Stato del cursore: K, L, F o G.
N 2	16391	PPC	Numero della linea di programma in esecuzione.
S 1	16393	VERSN	0 identifica la versione del nuovo BASIC.
S 2	16394	E-PPC	Numero della linea sulla quale sta il puntatore.
SX2	16396	D-FILE	Vedere organizzazione memoria.
S 2	16398	DF-CC	Indirizzo della posizione di stampa nella memoria di schermo.
SX2	16400	VARS	Vedere organizzazione memoria.
SN2	16402	DEST	Indirizzo della variabile in fase di assegnazione.
SX2	16404	E-LINE	Vedere organizzazione memoria.
SX2	16406	CH-ADD	Indirizzo del prossimo carattere da interpretare usato per PEEK, POKE e NEWLINE.
S 2	16408	X-PTR	Indirizzo del carattere che precede lo stato S del cursore.
SX2	16410	STKBOT	Vedere organizzazione memoria.
SX2	16412	STKEND	Vedere organizzazione memoria.
SN1	16414	BERG	Registro B.
SN2	16415	MEM	Indirizzo area usata come memoria per calcoli. A volte uguale MEMBOT.
S 1	16417		Non usato.
SX1	16418	DF-SZ	Numero delle linee della parte bassa dello schermo (compresa una linea bianca).
S 2	16419	S-TOP	Numero delle linee di programma del=la parte alta dello schermo durante la lista automatica.

Note	Indirizzo	Nome	Contenuto
SN2	16421	LAST-K	Ultimo tasto premuto.
SN1	16423		Stato della tastiera.
SN1	16424	MARGIN	Numero di linee bianche sopra o sotto il disegno (55).
SX2	16425	NXTLIN	Indirizzo della prossima linea di programma da eseguire.
S 2	16427	OLDPPC	Numero di linea da cui deve partire CONT.
SN1	16429	FLAGX	Flags per usi vari.
SN2	16430	STRLEN	Lunghezza della stringa in fase di assegnazione.
SN2	16432	T-ADDR	Indirizzo dell' elemento seguente nella tabella sintattica.
S 2	16434	SEED	Punto di partenza per RND. Viene preparato da RAND.
S 2	16436	FRAMES	Contatore dei fotogrammi dello schermo. Il bit 15 e' 1, i bit da 0 a 14 sono decrementati per ogni fotogramma. Esso puo' essere usato come Timer. PAUSE lo usa mettendo a 0 il bit 15 e ponendo nei bit da 0 a 14 la lunghezza della pausa. Quando il conto all'indietro e' arrivato a 0 la pausa termina. Se si interrompe la pausa con un qualunque tasto il bit 15 viene rimesso a 1.
S 1	16438	COORDS	Coordinata x dell'ultimo punto ot= tenuto con PLOT.
S 1	16439		Lo stesso per y.
S 1	16440	PR-CC	Byte meno significativo dell'indi= rizzo della prossima posizione per LPRINT in PRBUFF.
SX1	16441	S-POSN	Numero della colonna per la posi= zione di PRINT.
SX1	16442		Numero della linea per PRINT.
S 1	16443	CDFLAG	Flags per usi vari. Il bit 7 e' a 1 durante i calcoli e le fasi di stampa al video.
S 33	16444	PEBUFF	Buffer di stampa, 32 caratteri + il carattere NEWLINE.
SN30	16447	MEMBOT	Area di memoria per calcoli. Viene usata per memorizzare quei numeri che non possono essere posti nella STACK area.
S 2	16507		Non usati.

APPENDICE C

S C H E D A   B A S I C   ZX80

VARIABILI:

Intere:	primo carattere alfabetico, caratteri successivi o cifre o lettere senza spazi, contenuto numeri interi compresi tra -32768 e +32767.
Stringhe:	nome formato da una lettera seguita da $(dollaro), non c'e' limite al numero dei caratteri contenuti.

COSTANTI:

Intere:	numeri compresi tra -32768 e +32767.
Stringhe:	delimitate dagli apici, lunghezza a piacere, possono contenere qualunque carattere salvo gli apici.

VARIABILI CON INDICE:

Intere:	nome formato da una sola lettera, un solo indice e come indice una espressione intera.

VARIABILI DI CONTROLLO:

Intere:	nome formato da una sola lettera.

ESPRESSIONI ARITMETICHE:

Operatori aritmetici:

** (elevato a)
- (unitario)
* prodotto
/ divisione
+ somma (somma e sottrazione non hanno
- sottrazione ordine di precedenza tra loro)

Uso delle parentesi

ordine di valutazione da sinistra a destra con la precedenza con la quale sono stati listati gli operatori.

ESPRESSIONI RELAZIONALI:	Operatori relazionali:	= > < (senza ordine di precedenza tra loro). Valore -1 per condizione vera; 0 per condizione falsa.
	Operatori logici:	NOT,AND,OR (le precedenze sono quelle date dalla lista).
ESPRESSIONI BOOLEANE:	usano gli operatori	logici.
ISTRUZIONI:	NEW	inizializza il calcolato=re e cancella la memoria.
	LOAD	carica programmi e dati da nastro magnetico.
	SAVE	memorizza programmi e da=ti su nastro magnetico.
	RUN	manda in esecuzione il programma azzerando le variabili.
	RUNn	come sopra ma con parten=za dalla linea n.
	CONTINUE	fa continuare da n se n e' nel messaggio del si=stema, fa continuare da n+1 dopo uno STOP.
	REM	commenti a scopo documen=tativo.
	IF n THEN istruz.	esegue istruzione se la condizione n e' vera.
	INPUT dest	legge in dest il dato.
	PRINT lista	stampa il contenuto di lista, separatori di cam=po <;> e <,>.
	LIST n	lista il programma con il puntatore di linea ad n.
	LIST	lista il programma dallo inizio.

STOP	ferma il programma, per continuare CONTINUE.
DIM A(n)	predispone una variabile numerica con indice for=mata da n+1 elementi.
FOR K = n1 TO n2	gestisce con il contatore K un ciclo per valore di K = n1 e valore finale di K = n2 dando ad ogni giro l'incremento di 1 a K.
GOTO n	salta alla linea n.
POKE n1, n2	scrive all'indirizzo n1 il valore n2.
RANDOMISE n	pone l'inizio per la ge=nerazione dei numeri a caso al valore n.
RANDOMISE	come sopra, ma n = valore del contatore dei foto=grammi dello schermo.
CLEAR	cancella tutte le varia=bili.
CLS	azzera la parte superiore dello schermo.
GOSUB n	come GOTO ma conserva nello STACK l'indicazione per ritornare al program=ma principale.
RETURN	fa prelevare dallo STACK l'indicazione per tornare al programma principale.
NEXT K	chiude il ciclo iniziato da FOR, incrementa K e ne controlla il valore.
LET	consente di fare qualun=que operazione di asse=gnazione o di calcolo.

Esiste anche il tasto BREAK per interrompere l'esecuzione di un programma se non e' in at=

tesa di INPUT.

FUNZIONI IMPLEMENTATE:		
	RND(n)	genera un numero pseudo-random minore o uguale a n. La sequenza e' influ= enzata nel punto di par= tenza da RANDOMISE.
	ABS(espress.)	fornisce il valore asso= luto dell'espressione.
	PEEK(n)	fornisce il contenuto del byte di memoria di indi= rizzo n.
	USR(n)	permette di andare ad e= seguire un codice macchi= na memorizzato a partire da n.
	CHR$(x)	fornisce il carattere corrispondente al codice numerico x.
	TL$(stringa)	ritorna la stringa senza il primo carattere.
	CODE(stringa)	fornisce il codice nume= rico del primo carattere della stringa.
	STR$(x)	fornisce una stringa di caratteri corrispondente al numero x.

APPENDICE D

SCHEDA BASIC NUOVA ROM E ZX81

VARIABILI.

Numeriche	Nome: primo carattere alfabetico, altri cifre o lettere o spazi, quanti si vuole. Numeri interi e decimali con precisione tra 9 e 10 cifre e grandezza compresa tra 10 ele= vato a -39 e 10 elevato a +38.
Stringhe	Nome formato da una lettera seguita da $. Non esiste limite al numero dei caratteri.

COSTANTI.

Numeriche	Stesse possibilita' che per i contenuti del= le variabili numeriche.
Stringhe	Delimitate da apici, lunghezza a piacere, possono contenere qualunque carattere salvo gli apici. La stringa nulla e' "". Per otte= nere gli apici stampabili all'interno di una stringa si deve usare il carattere "doppio apice" o CHR$(192).

VARIABILI CON INDICE.

Numeriche	Nome formato da una sola lettera, indici multipli, contenuti come per le variabili numeriche semplici. Si puo' usare lo stesso nome gia' usato per una variabile semplice.
Stringhe	Nome formato da una lettera seguita da $, indici multipli, contenuti come per le stringhe semplici, tutti gli elementi devono avere lo stesso numero di caratteri. Il nome usato per una stringa con indici non puo' essere usato per una stringa senza indici.

Gli indici possono essere costanti, variabili numeriche o espressioni numeriche e vengono arrotondati all'intero piu' prossimo.

VARIABILI DI CONTROLLO.

Numeriche	Nome formato da una sola lettera. Sono usate per controllare i cicli FOR/NEXT e all'interno della variabile viene memorizzato il numero della linea della prima istruzione del ciclo.

ESPRESSIONI.

Operatori aritmetici:	**	elevato a. Esempio: X**Y, si ha errore B se X negativo. Priorita' 10.
	-	unitario, segno del numero. Priorita' 9.
	*,/	moltiplicato, diviso. Priorita' 8.
	+,-	addizione, sottrazione. Priorita' 6.

Operatori relazionali:	=	uguale.	Priorita' 5.
	>	maggiore.	" "
	<	minore.	" "
	<=	min. o ug..	" "
	>=	magg. o ug..	" "
	<>	diverso.	" "

Operatori logici:	NOT	negazione. Priorita' 4.
	AND	prodotto logico. Priorita' 3.
	OR	somma logica. Priorita' 2.

Gli operatori relazionari e gli operatori logici produco una variabile logica di valore:

1 se condizione vera;
0 se condizione falsa.

Le espressioni logiche e relazionali possono far parte di espressioni aritmetiche, ad esse viene sostituito il valore della variabile logica. Le espressioni vengono valutate da sinistra a destra tenendo conto delle parentesi e delle priorita'.

FRASI BASIC.

Nella descrizione delle frasi si usano le seguenti convenzioni:

a	rappresenta una singola lettera;
v	rappresenta una variabile;
x,y,z	rappresentano espressioni numeriche;
m,n	rappresentano espressioni numeriche arrotondate all'intero piu' vicino;
e	rappresenta una espressione;
f	rappresenta una espressione stringa;
s	rappresenta una frase BASIC.

Ricordiamo che:

- Si possono usare dovunque espressioni, salvo che per i numeri di linea del programma.
- Tutte le frasi possono essere usate sia in modo im= mediato che differito (anche se questo puo' non avere molto significato in alcuni casi) salvo la INPUT che puo' solo essere usata in modo differito.

Comandi	Commento
CLEAR	Cancella tutte le variabili liberando lo spazio che occupavano.
CLS	Pulisce lo schermo, cioe' pone spazi nella memoria di schermo.
CONT	Se il codice di errore e' p/q e q <> 0, CONT fa eseguire un: GOTO q se p<>9 GOTO q+1 se p=9.
COPY	Manda sulla stampante, se collegata, una copia dello schermo. Se la stampante non e' collegata non ha alcun effetto.
DIMa(n1,..,nk)	Cancella una variabile con indice di nome "a" e la ridefinisce. Non da' errore di ridi= mensionamento. Tutti gli elementi vengono inizializzati a 0. Errore 4 se manca spazio. Puo' esistere una variabile singola di nome "a".
DIMa$(n1,.,nk)	Cancella una variabile stringa con indice avente lo stesso nome e la ridefinisce. L'ultimo dato in parentesi non e' una dimensione, ma la lunghezza di ogni elemento in caratteri. Tutti gli elementi vengono inizializzati con il carattere spazio. Errore 4 se manca spazio. Non puo' esistere una

variabile stringa singola di nome "a$".

FORa=xTOy — Significa: FORa=xTOySTEP1.

FORa=xTOySTEPz — Cancella, se esiste, la variabile singola di nome "a" e crea una variabile di controllo di nome "a", x e'il valore iniziale di a, y e' il valore finale di a, z e' l'incremento da usare ad ogni ciclo. L'indirizzo della prima istruzione del ciclo e' quello della linea dopo il FOR se lavora in modo differito, della linea precedente il FOR se lavora in modo immediato.
Se x>y e z>=0 oppure se x<y e z<=0 salta alla linea del NEXTa. Errore 4 se manca spazio per la variabile di controllo.

GOSUBn — Pone il numero della linea del GOSUB nella Stack area e poi salta alla linea n. Errore 4 se non trova il relativo RETURN.

GOTOn — Salta alla linea n, se la linea n manca, salta alla prima linea con numero > n.

IFxTHENs — Se la condizione x e' vera (variabile logica uguale a 1) esegue l'istruzione s, altrimenti prosegue dalla linea seguente.

INPUTv — Si ferma in attesa di dati con il cursore a L per dati numerici e ad L tra apici per stringhe. Se si risponde premendo il tasto STOP e si e' in attesa di numeri il programma si ferma con errore D. Se si risponde con il tasto STOP all'attesa di stringa viene registrata la parola STOP. Se si usa in modo immediato si ha errore 8. I dati ricevuti in INPUT non restano sul video.

LETv=e — La parola chiave LET e' obbligatoria. Una variabile singola non e' definita fino a quando non compare in una LET a sinistra di un = o in una frase INPUT. Se v e' una variabile stringa con indice o una porzione di stringa (sliced), cioe' una variabile stringa di dimensioni predeterminate, vengono troncati a destra i caratteri eccedenti o aggiunti spazi di riempimento.

LIST — Corrisponde a LISTO.

LISTn — Lista il programma sul video a partire dalla linea n. Errore 4 o 5 se la lista non entra nello schermo.

LLIST — Corrisponde a LLISTO.

LLISTn — Come LIST, ma la lista va alla stampante, se la stampante non e' collegata non agisce.
Se si usa BREAK da' errore D.

LOADf — Cerca un programma di nome f sul nastro e lo carica in memoria insieme alle sue variabili.
SE f e' la stringa nulla, carica il primo programma che trova sul nastro.
Se si preme BREAK o se si ha un errore sul nastro si ha:
1) se non e' ancora stato letto un programma si ferma con errore D;
2) se e' stato letto un pezzo di programma esegue automaticamente un NEW.

LPRINT... — Come il comando PRINT, ma invia i dati alla stampante. Viene inviata una linea quando:
1) si passa da una linea alla seguente;
2) un comando non termina con "," o ";";
3) una , o un TAB richiede una nuova linea;
4) alla fine del programma rimane qualcosa da stampare.
Il comando AT ha significato solo riguardo al numero di colonna. Se si preme BREAK da' errore D. Effetto nullo senza la stampante.

NEW — Cancella il programma e le variabili, ma non tocca la parte di memoria dopo l'indirizzo contenuto in RAMTOP.

NEXTa — 1) Cerca la variabile di controllo a;
2) Aggiunge alla variabile lo STEP;
3) Se STEP>=0 e a> limite o
se STEP<=0 e a< limite salta alla prima linea del ciclo.
Errore 1 se a non e' una variabile di controllo. Errore 2 se la variabile a non esiste del tutto.

PAUSEn — Sospende il lavoro per una durata pari al=l'emissione di n fotogrammi (50 fotogram=mi al secondo) o fino a quando viene premuto un qualunque tasto. Se non e' 0<=n<=65535 si ha errore B. Se n>=32767 si puo' interrompere la pausa solo premendo un tasto.

PLOTm,n — Scrive il puntino di coordinate m,n e sposta la posizione di stampa dopo il puntino.
0<=m<=63 e 0<=n<=43, altrimenti errore B.

POKEm,n — Scrive il valore n nel byte m. Deve essere:

0<=m<=65535 e -255<=n<=255, altrimenti si ha errore B.

PRINT.... I "..." stanno per la lista di elementi da stampare. Gli elementi possono essere sepa= rati da "," o da ";". Il ";" non modifica la posizione di stampa, mentre la "," sposta la posizione di stampa di 16 posizioni almeno, cioe' fa posizionare o in colonna 0 o in co= lonna 16. Se la lista di stampa non termina con "," o ";" la posizione di stampa si sposta all'inizio della linea seguente.
Gli elementi da stampare possono essere:
1) stringa nulla e quindi niente;
2) una espressione numerica. Viene stampato il segno meno se il valore e' negativo. Se il valore assoluto del numero da stam= pare e' <=(10**(-5)) o >=(10**13) esso viene stampato usando la notazione espo= nenziale. La mantissa viene stampata con al massimo 8 cifre ed il punto decimale dopo la prima. L'esponente viene dopo E, il segno ed e' formato da 1 o 2 cifre. Se il numero e' compreso nell'intervallo esso viene stampato con la consueta no= tazione decimale e con al massimo 8 cifre significative.
3) una espressione stringa. Le parole chiave del linguaggio vengono espanse, il carat= tere "quote image" viene stampato come un doppio apice. I caratteri che non hanno corrispondenza in stampa vengono stampati come punti interrogativi.
4) ATm,n. Essa agisce sulla posizione di stampa, la linea viene contata a partire dall'alto, la colonna a partire da sini= stra. Deve essere: 0<=m<=21, altrimenti si ha errore 5, ma se m=22 o m=23 errore B; 0<=n<=31, altrimenti errore B.
5) TABn. Si considera n modulo 32. Viene mo= dificata la posizione di stampa sulla stessa linea, a meno che questo non com= porti spostamenti all'indietro, nel qual caso si passa sulla prossima linea. Deve essere 0<=n<=255, altrimenti errore B.

Se si hanno solo 3K o meno di memoria si ha errore 4 (OUT OF MEMORY).
Errore 5 significa che lo schermo e' pieno. In questi due casi CONT consente di procedere dopo aver svuotato lo schermo.

RAND Corrisponde a RANDO.

RANDn — Inizializza la variabile, chiamata SEED, che il sistema usa per generare i numeri pseudo random con la funzione RND. Se n <> 0 viene posta SEED=n; se n=0 viene posta SEED uguale al valore di un'altra variabile del sistema, chiamata FRAMES, ed e' il contatore dei fotogrammi dello schermo. Si ha errore B se n non e' compreso nell'intervallo 0-65535.

REM... — Serve per i commenti, "..." puo' contenere qualunque carattere meno NEWLINE.

RETURN — Preleva un numero di linea dall'area STACK dei GOSUB e salta a quella linea.
Si ha errore 7 se l'area stack e' vuota.

RUN — Corrisponde a RUN0.

RUNn — Esegue un CLEAR automatico e fa saltare alla linea n. Se non si vule il CLEAR si deve usare GOTOn.

SAVEf — Memorizza un programma e le sue variabili sul nastro con il nome f. Non si puo' usare SAVE all'interno di un sottoprogramma.
Si ha errore F se f e' la stringa nulla.

SCROLL — Fa scorrere lo schermo di una linea verso l'alto, perdendo la linea piu' in alto e liberandone una in basso. La linea liberata contiene come primo carattere NEWLINE.

STOP — Fa fermare il programma con codice di errore 9. CONT fa proseguire dalla linea seguente.

TO — Questa parola chiave fa parte del comando FOR/NEXT e viene usata in questo modo per ottenere le substringhe.
Si scrive f(m TO n) per indicare quella parte di stringa f che e' compresa tra il carattere di posto m e quello di posto n.
I due numeri m ed n devono essere positivi altrimenti si ha errore 3. Si espongono con degli esempi i casi possibili:

```
"BELLO"( TO5) da' "BELLO"
"BELLO"(2TO ) da' "ELLO"
"BELLO"( TO ) da' "BELLO"
"BELLO"(2TO2) da' "E"
"BELLO"(3TO8) da' errore
"BELLO"(5TO4) da' "" stringa nulla.
```

UNPLOTm,n — Agisce come PLOT, ma cancella il puntino.

FUNZIONI:

Per le funzioni che richiedono un argomento questo puo' anche essere una espressione. Se l'argomento e' una espressione esso deve essere racchiuso tra parentesi, se e' una costante o una variabile non e' necessario fare uso delle parentesi. L'operando viene indicato con x e si specifica il tipo.

Funz.	Operando	Risultato
ABS	numero	Valore assoluto.
ACS	numero	Arcocoseno in radianti. Errore A se non e' -1<=x<=1.
ASN	numero	Arcoseno in radianti. Errore A se non e' -1<=x<=1.
AT	vedere comando PRINT.	
ATN	numero	Arcotangente in radianti.
CHR$	numero	Il carattere di codice x arrotondato all'in= l'intero piu' vicino. Errore B se non e' 0<=x<=255.
CODE	stringa	Il codice del primo carattere di x o 0 se x e' la stringa nulla.
COS	numero	Coseno. L'operando deve essere in radianti.
EXP	numero	Il numero "e" elevato a x.
INKEY$		(nessun argomento) Legge dalla tastiera il carattere corrispondente al tasto premuto con il cursore nello stato L, se non si preme alcun tasto da' la stringa nulla.
INT	numero	Parte intera del numero troncato.
LEN	stringa	Lunghezza in caratteri della stringa.
LN	numero	Logaritmo naturale (in base "e") di x. Errore A se x<=0.
NOT	vedere operatori logici.	
PEEK	numero	Il valore del byte di indirizzo x, arro= tondato al piu' vicino intero. Errore B se

non e' 0<=x<=255.

PI — (nessun argomento) Il valore di "pigreco", 3.14159265.

RND — (nessun argomento) Il prossimo numero della sequenza dei numeri pseudo random generati. Il numero generato e' compreso tra 0 e 1.

SGN numero — Segno del numero: -1,0,1.

SIN numero — Seno. L'operando deve essere in radianti.

SQR numero — Radice quadrata. Errore B se x<0.

STR$ numero — La stringa di caratteri corrispondente alle cifre del numero con segno se negativo.

TAB — vedere il comando PRINT.

TAN numero — Tangente. L'operando deve essere in radianti.

USR numero — Va ad eseguire il programma in codice mac= china memorizzato in x (arrotondato all'inte= ro piu' vicino). Al ritorno il risultato si trova nei registri BC. Errore B se non e' 0<=x<=65535.

VAL stringa — Valuta x come espressione numerica. Errore C se la stringa non e' numerica.

APPENDICE E

ERRORI SEGNALATI DAL SISTEMA

ZX80

Il sistema segnala gli errori facendo apparire nella parte bassa dello schermo a sinistra un codice nella forma n/m dove : n=numero dell'errore m=numero di linea del programma che ha generato la segnalazione.

TABELLA DEGLI ERRORI

Cod.	Significato	Situazione
0	Si e' usato il tasto BREAK ; m rappresenta il numero della linea dopo quella in ese= cuzione al momento del BREAK.Se m=-1 oppure m=-2 e'stato eseguito comando in modo im= mediato. Puo' essere m negativo oppure m un numero di linea non presente nel programma; e' stato eseguito un GOTO m. Alla fine del programma m rappresenta l'ultimo numero di linea presente nel programma.	varie
1	m=numero di linea che ha causato l'errore. Esiste un NEXT con una variabile, gia' de= fita dal programma ,ma che non e' la stessa usata nel FOR precedente il NEXT, m= numero della linea che ha causato l'errore.	NEXT
2	E'stata usata una variabile non definita in precedenza. Una variabile singola viene de= finita con una LET di assegnazione. Una variabile con indice viene definita mediante la DIM, m = numero della linea che ha causato l'errore.	varie
3	L'indice di una variabile con indice e' fuori dai limiti definiti dalla DIM o c'e' errore nel calcolo dell'indice, m = numero della linea che ha causato l'errore.	varie
4	Non c'e' piu' posto per aggiungere una nuova variabile numerica o per aumentare il numero dei caratteri di una stringa o manca posto sullo schermo.	LET INPUT DIM PRINT
5	Non c'e' piu' posto sullo schermo. Se in questo caso si preme CONT due volte e poi NEW LINE la stampa continua, m=linea che	PRINT

	ha causato l'errore.	
6	Si e' avuto supero di capacita' durante il calcolo, cioe' il risultato e' minore di -32768 o maggiore di +32767. A volte si ha questo errore anche per risultato =-32768; m=numero di linea che ha causato l'errore.	varie
7	Si e' incontrato un RETURN senza che sia stato preceduto da un GOSUB, m=-2.	RETURN
8	Si e' tentato di usare l'istruzione INPUT in modo immediato.	INPUT
9	m=numero di linea contenente il comando STOP. Usando CONT il programma continua dalla linea seguente la m.	STOP

Dopo una segnalazione di errore da parte del sistema, a seconda dei casi si interverra' opportunamente, eventualmente modificando il programma.

NUOVA ROM E ZX81

Il sistema al termine di ogni lavoro e quando incontra alcune istruzioni particolari segnala lo stato in cui si trova mediante un messaggio che appare nell'angolo in basso a sinistra dello schermo. Abitualmente questo messaggio viene chiamato "messaggio di errore", in realta' sarebbe piu' corretto chiamarlo "messaggio di stato", dato che quello che viene segnalato non sempre e' un errore.

Il messaggio si compone di due parti: n/m.

Dove:
- n e' il numero della linea dove si e' fermato il pro= gramma
- m e' il numero distintivo del messaggio in esadecimale cioe' un numero da 0 a F.

TABELLA DEI MESSAGGI

Cod.	Significato	Situazione
0	Tutto e' andato bene oppure salto ad una linea con numero maggiore di tutte quelle esistenti. Se si usa CONT in modo immediato il programma prosegue dalla linea n.	Varie
1	La variabile di controllo non esiste, cioe' non e' stata citata nel FOR precedente il NEXT, ma esiste come variabile ordinaria.	NEXT

2	Si e' usata una variabile che non era stata definita precedentemente. Se la variabile e' singola non c'e' stata una frase: LET var.= espressione o INPUT var.. Se la variabile e' con indice non c'e' la frase di dimensionamento DIM. Se la variabile e' di controllo, essa non stata citata nel FOR e non esiste come variabile ordinaria.	Varie
3	Indici fuori dal range stabilito. Se oltre ad essere fuori range l'indice e' negativo o >65535 si ha errore di codice B.	Varia=bili con indice
4	Manca spazio in memoria. Il numero della linea nel messaggio puo' essere incompleto proprio a causa della mancanza di memoria. Si puo' avere un programma errato che usa troppa memoria nell'area STACK.	LET,INPUT, DIM,PRINT, LIST,PLOT, UNPLOT, FOR,GOSUB, calcolo di funzioni complica=te.
5	Non si ha piu' spazio sul video. Se si usa CONT lo schermo si libera e il lavoro puo' proseguire.	PRINT,LIST
6	Supero di capacita' (overflow) durante un un calcolo (risultato in valore assoluto > 10**38).	Calcoli
7	Incontra un RETURN, ma non c'e' stato prima un GOSUB.	RETURN
8	Si e' tentato di usare il comando INPUT in modo immediato.	INPUT
9	E' stato eseguito un comando STOP. Se si usa CONT il programma non riesegue la li=nea del comando STOP, ma prosegue.	STOP
A	Argomento non valido nel calcolo di una funzione.	SQR,LN, ASN,ACS.
B	Numero intero fuori dal range. Se il co=mando richiede un numero intero, esso viene ottenuto arrotondando il numero decimale in questione all'intero piu' vicino e in questo modo si esce dal range.	RUN,RAND, POKE,DIM, GOTO,LIST, GOSUB, LLIST, PAUSE,

		PLOT,USR UNPLOT, CHR$,PEEK. Variabili con ind.
C	Si usa una VAL con stringa non numerica.	VAL
D	1) Programma interrotto dal tasto BREAK.	Alla fine di ogni frase o in LOAD,SAVE, LPRINT, LLIST, COPY.
	2) Il dato di risposta ad un INPUT nume= rico inizia con STOP. In questo modo si puo' interrompere un programma durante l'INPUT.	INPUT
E	Non usato.	
F	Il nome del programma usato in SAVE e' la stringa nulla.	SAVE

APPENDICE F

IL LINGUAGGIO MACCHINA

Si riporta una tabella contenente le istruzioni in linguaggio macchina, la traduzione in esadecimale e decimale ed una breve spiegazione del significato di ogni istruzione. Si noti che i valori decimali vanno da 0 a 255 e quindi, cio' che, se si lavora in assoluto, viene interpretato come una istruzione in linguaggio macchina, se si lavora in Basic ha un significato completamente diverso. Il contenuto dei byte e' il medesimo, ma quello che cambia e' la loro interpretazione.

Assembler	Cod. Macchina Esad.	Decim.	Commento
NOP	00	0	nessuna operazione
LD BC,nn	01	1	il numero nn va in BC
LD (BC),A	02	2	il contenuto di A va nel byte puntato da BC
INC BC	03	3	incrementa BC di 1
INC B	04	4	incrementa B di 1
DEC B	05	5	decrementa B di 1
LD B,n	06	6	il numero n va in B
RLCA	07	7	rotazione circolare sinistra dell' accumulatore
EX,AF,AF'	08	8	scambia AF con AF'
ADD HL,BC	09	9	somma al contenuto HL quello di BC
LD A,(BC)	0A	10	il contenuto del byte puntato da BC va in A
DEC BC	0B	11	decrementa BC di 1
INC C	0C	12	incrementa C di 1
DEC C	0D	13	decrementa C di 1
LD C,n	0E	14	il numero n va in C
RRCA	0F	15	rotazione circolare destra dell' accumulatore
DJNZ disp	10	16	decrementa B e salta se B e' diverso da 0
LD DE,nn	11	17	il numero nn va in DE
LD (DE),A	12	18	il contenuto di A va nel byte puntato da DE
INC DE	13	19	incrementa DE di 1
INC D	14	20	incrementa D di 1
DEC D	15	21	decrementa D di 1
LD D,n	16	22	il numero n va in D

RLA	17	23	rotazione sinistra dell'ac= cumulatore
JR disp	18	24	salto relativo incondizionato
ADD HL,DE	19	25	somma al contenuto di HL quello di DE
LD A,(DE)	1A	26	il contenuto del byte puntato da DE va in A
DEC DE	1B	27	decrementa DE di 1
INC E	1C	28	incrementa E di 1
DEC E	1D	29	decrementa E di 1
LD E,n	1E	30	il numero n va in E
RRA	1F	31	rotazione destra dell'accu= mulatore
JR NZ,disp	20	32	se Z = 1 continua, se Z = 0 PC=PC+disp
LD HL,nn	21	33	il numero nn va in HL
LD(nn),HL	22	34	H va in (nn+1), L va in (nn)
INC HL	23	35	incrementa HL di 1
INC H	24	36	incrementa H di 1
DEC H	25	37	decrementa H di 1
LD H,n	26	38	il numero n va in H
DAA	27	39	converte in BCD il risultato
JR Z,disp	28	40	se Z=0 continua, se Z=1 PC=PC+disp
ADD HL,HL	29	41	moltiplica per 2 il contenuto di HL
LD HL,(nn)	2A	42	il contenuto del byte (nn) va in HL
DEC HL	2B	43	decrementa HL di 1
INC L	2C	44	incrementa L di 1
DEC L	2D	45	decrementa L di 1
LD L,n	2E	46	il numero n va in L
CPL	2F	47	complementa a 1 i bits di A
JR NC,disp	30	48	se C=1 continua, se C=0 PC=PC+disp
LD SP,nn	31	49	il numero nn va in SP
LD (nn),A	32	50	il contenuto di A va nel byte (nn)
INC SP	33	51	incrementa SP di 1
INC (HL)	34	52	incrementa di 1 il contenuto del byte (HL)
DEC (HL)	35	53	decrementa di 1 il contenuto del byte (HL)
LD (HL),n	36	54	il numero n va nel byte (HL)
SCF	37	55	pone a 1 il flag di CARRY
JR C,disp	38	56	se C = 0 continua, se C = 1 PC=PC+disp
ADD HL,SP	39	57	somma al contenuto di HL quello di SP
LD A,(nn)	3A	58	il contenuto del byte (nn) va in A

DEC SP	3B	59	decrementa SP di 1
INC A	3C	60	incrementa A di 1
DEC A	3D	61	decrementa A di 1
LD A,n	3E	62	il numero n va in A
CCF	3F	63	complementa a 1 il flag di CARRY
LD B,B	40	64	carica B in B
LD B,C	41	65	carica C in B
LD B,D	42	66	carica D in B
LD B,E	43	67	carica E in B
LD B,H	44	68	carica H in B
LD B,L	45	69	carica L in B
LD B,(HL)	46	70	il contenuto del byte (HL) va in B
LD B,A	47	71	carica A in B
LD C,B	48	72	carica B in C
LD C,C	49	73	carica C in C
LD C,D	4A	74	carica D in C
LD C,E	4B	75	carica E in C
LD C,H	4C	76	carica H in C
LD C,L	4D	77	carica L in C
LD C,(HL)	4E	78	il contenuto del byte (HL) va in C
LD C,A	4F	79	carica A in C
LD D,B	50	80	carica B in D
LD D,C	51	81	carica C in D
LD D,D	52	82	carica D in D
LD D,E	53	83	carica E in D
LD D,H	54	84	carica H in D
LD D,L	55	85	carica L in D
LD D,(HL)	56	86	il contenuto del byte (HL) va in D
LD D,A	57	87	carica A in D
LD E,B	58	88	carica B in E
LD E,C	59	89	carica C in E
LD E,D	5A	90	carica D in E
LD E,E	5B	91	carica E in E
LD E,H	5C	92	carica H in E
LD E,L	5D	93	carica L in E
LD E,(HL)	5E	94	il contenuto del byte (HL) va in E
LD E,A	5F	95	carica A in E
LD H,B	60	96	carica B in H
LD H,C	61	97	carica C in H
LD H,D	62	98	carica D in H
LD H,E	63	99	carica E in H
LD H,H	64	100	carica H in H
LD H,L	65	101	carica L in H
LD H,(HL)	66	102	il contenuto del byte (HL) va in H
LD H,A	67	103	carica A in H
LD L,B	68	104	carica B in L

LD L,C	69	105	carica C in L
LD L,D	6A	106	carica D in L
LD L,E	6B	107	carica E in L
LD L,H	6C	108	carica H in L
LD L,L	6D	109	carica L in L
LD L,(HL)	6E	110	il contenuto del byte (HL) va in L
LD L,A	6F	111	carica A in L
LD (HL),B	70	112	carica B nel byte (HL)
LD (HL),C	71	113	carica C nel byte (HL)
LD (HL),D	72	114	carica D nel byte (HL)
LD (HL),E	73	115	carica E nel byte (HL)
LD (HL),H	74	116	carica H nel byte (HL)
LD (HL),L	75	117	carica L nel byte (HL)
HALT	76	118	HALT per la CPU
LD (HL),A	77	119	carica A nel byte (HL)
LD A,B	78	120	carica B in A
LD A,C	79	121	carica C in A
LD A,D	7A	122	carica D in A
LD A,E	7B	123	carica E in A
LD A,H	7C	124	carica H in A
LD A,L	7D	125	carica L in A
LD A,(HL)	7E	126	il contenuto del byte (HL) va in A
LD A,A	7F	127	carica A in A
ADD A,B	80	128	somma B ad A
ADD A,C	81	129	somma C ad A
ADD A,D	82	130	somma D ad A
ADD A,E	83	131	somma E ad A
ADD A,H	84	132	somma H ad A
ADD A,L	85	133	somma L ad A
ADD A,(HL)	86	134	somma (HL) ad A
ADD A,A	87	135	moltiplica per 2 il contenuto di A
ADC A,B	88	136	somma ad A il contenuto di B + il CARRY
ADC A,C	89	137	somma ad A il contenuto di C + il CARRY
ADC A,D	8A	138	somma ad A il contenuto di D + il CARRY
ADC A,E	8B	139	somma ad A il contenuto di E + il CARRY
ADC A,H	8C	140	somma ad A il contenuto di H + il CARRY
ADC A,L	8D	141	somma ad A il contenuto di L + il CARRY
ADC A,(HL)	8E	142	somma ad A il contenuto di (HL) + il CARRY
ADC A,A	8F	143	molt. A per 2, risult.+ CARRY

SUB B	90	144	sottrae ad A il contenuto di B
SUB C	91	145	sottrae ad A il contenuto di C
SUB D	92	146	sottrae ad A il contenuto di D
SUB E	93	147	sottrae ad A il contenuto di E
SUB H	94	148	sottrae ad A il contenuto di H
SUB L	95	149	sottrae ad A il contenuto di L
SUB (HL)	96	150	sottrae ad A il contenuto di (HL)
SUB A	97	151	sottrae ad A il contenuto di A
SBC A,B	98	152	A = A - B - CARRY
SBC A,C	99	153	A = A - C - CARRY
SBC A,D	9A	154	A = A - D - CARRY
SBC A,E	9B	155	A = A - E - CARRY
SBC A,H	9C	156	A = A - H - CARRY
SBC A,L	9D	157	A = A - L - CARRY
SBC A,(HL)	9E	158	A = A - (HL) - CARRY
SBC A,A	9F	159	A = A - A - CARRY
AND B	A0	160	A = AND logico tra A e B, mod. flags
AND C	A1	161	A = AND logico tra A e C, mod. flags
AND D	A2	162	A = AND logico tra A e D, mod. flags
AND E	A3	163	A = AND logico tra A e E, mod. flags
AND H	A4	164	A = AND logico tra A e H, mod. flags
AND L	A5	165	A = AND logico tra A e L mod. flags
AND (HL)	A6	166	A = AND logico tra A e (HL) mod. flags
AND A	A7	167	A = AND logico tra A e se' stesso, mod. flags
XOR B	A8	168	A = XOR tra A e B, mod. flags
XOR C	A9	169	A = XOR tra A e C, mod. flags
XOR D	AA	170	A = XOR tra A e D mod. flags
XOR E	AB	171	A = XOR tra A e E mod. flags
XOR H	AC	172	A = XOR tra A e H, mod. flags
XOR L	AD	173	A = XOR tra A e L, mod. flags
XOR (HL)	AE	174	A = XOR tra A e (HL), mod. flags
XOR A	AF	175	A = XOR tra A e se' stesso, mod. flags
OR B	B0	176	A = OR tra A e B, mod. flags
OR C	B1	177	A = OR tra A e C, mod. flags

OR D	B2	178	A = OR tra A e D, mod. flags
OR E	B3	179	A = OR tra A e E, mod. flags
OR H	B4	180	A = OR tra A e H, mod. flags
OR L	B5	181	esegue l'OR logico su L
OR (HL)	B6	182	esegue l'OR logico sul byte (HL)
OR A	B7	183	esegue l'OR logico su A
CP B	B8	184	sottrae B da A, mod. flags
CP C	B9	185	sottrae C da A, mod. flags
CP D	BA	186	sottrae D da A, mod. flags
CP E	BB	187	sottrae E da A, mod. flags
CP H	BC	188	sottrae H da A, mod. flags
CP L	BD	189	sottrae L da A, mod. flags
CP (HL)	BE	190	sottrae il byte (HL) da A, modifica i flags
CP A	BF	191	sottrae A da A, mod. flags
RET NZ	C0	192	se Z=0, return
POP BC	C1	193	BC e' caricato con gli ultimi due bytes dell'area STACK
JP NZ,nn	C2	194	se Z=0, PC=nn (salta a nn)
JP nn	C3	195	salto incondizionato assoluto PC=nn
CALL NZ,nn	C4	196	se Z=0, CALL nn
PUSH BC	C5	197	salva BC in due bytes dello STACK
ADD A,n	C6	198	somma il numero n ad A
RST 0h	C7	199	accesso allo stack per salto a sottoprogramma
RET Z	C8	200	se Z=1, return
RET	C9	201	return incondizionato
JP Z,nn	CA	202	se Z=1, salta a nn
Istruz. a 2 byte	CB	203	prefisso per operazioni sui bits
CALL Z,nn	CC	204	se Z=1, CALL nn
CALL nn	CD	205	chiamata subroutine
ADC A,n	CE	206	carica in A la somma A + A + CARRY
RST 8h	CF	207	accesso allo STACK per salto a sottoprogramma
RET NC	D0	208	se CARRY=0, return
POP DE	D1	209	D = (SP+1), E= (SP)
JP NC,nn	D2	210	se CARRY=0, PC = nn
OUT port,A	D3	211	A --> port
CALL NC,nn	D4	212	se CARRY=0, CALL nn
PUSH DE	D5	213	(SP-2)=E, (SP-1)=D
SUB n	D6	214	sottrae il numero n da A
RST 10h	D7	215	accesso allo STACK per salto a sottoprogramma
RET C	D8	216	se CARRY=1, return
EXX	D9	217	scambia i due set di registri
JP C,nn	DA	218	se CARRY=1, PC=nn
IN A, port	DB	219	port --> A

CALL C,nn	DC	220	se CARRY=1, CALL nn
Istruz. a 2 byte	DD	221	indirizzamento indicizzato con IX + disp
SBC A,n	DE	222	sottrae n ed il CARRY da A
RST 18h	DF	223	accesso allo STACK per salto a sottoprogramma
RET PO	E0	224	se la condizione e' vera, return
POP HL	E1	225	H = (SP+1), L = (SP)
JP PO,nn	E2	226	se la condizione e' vera, PC = nn
EX (SP),HL	E3	227	scambia H con (SP+1) e L con (SP)
CALL PO,nn	E4	228	se la condizione e' vera, CALL nn
PUSH HL	E5	229	(SP-2) = L, (SP-1) = H
AND n	E6	230	esegue l'AND del numero n con A
RST 20h	E7	231	accesso allo STACK per salto a sottoprogramma
RET PE	E8	232	se la condizione e' vera, return
JP (HL)	E9	233	PC = HL
JP PE,nn	EA	234	se la condizione e' vera, PC = nn
EX DE,HL	EB	235	scambio dei contenuti tra DE e HL
CALL PE,nn	EC	236	se la condizione e' vera, CALL nn
Istruz. a 2 byte	ED	237	prefisso per usi diversi
XOR n	EE	238	OR esclusivo del numero n con A
RST 28h	EF	239	accesso allo STACK per salto a sottoprogramma
RET P	F0	240	se P = 1, return
POP AF	F1	241	A = (SP+1), F = (SP)
JP P,nn	F2	242	se P = 1, PC = nn
DI	F3	243	0 --> IFF
CALL P,nn	F4	244	se P = 1, CALL nn
PUSH AF	F5	245	F = (SP-2), A = (SP-1)
OR n	F6	246	OR del numero n con A
RST 30h	F7	247	accesso allo STACK per salto a sottoprogramma
RET M	F8	248	se M = 1, return
LD SP,HL	F9	249	carica HL in SP
JP M,nn	FA	250	se M = 1, PC = nn
EI	FB	251	1 --> IFF
CALL M,nn	FC	252	se M = 1, CALL nn
Istruz. a 2 byte	FD	253	indirizzamento indicizzato con IY + disp
CP n	FE	254	sottrae ad A il valore n, modifica i flags

RST 38h	FF	255	accesso allo STACK per salto a sottoprogramma

CARATTERISTICHE

Registri generali con possibilita' di utilizzo a coppie:

Principali		Alternativi (memorie tampone)	
Accumulatore	Flag	Accumulatore	Flag
A	F	A'	F'
Utilita' generale			
B	C	B'	C'
D	E	D'	E'
H	L	H'	L'

Utilizzi speciali

I (interrupt)	R (refresh)
IX	(index doppia lunghezza)
IY	(index doppia lunghezza)
SP	(stack pointer)
PC	(program counter)

IL SISTEMA OPERATIVO DELLO ZX80

Si possono andare a leggere in ROM le routine del sistema operativo, servendosi della funzione PEEK. Poi con pazienza si cerca di passare dai codici decimali letti a quelli esadecimali e poi da questi alle istruzioni in assembler, oppure si passa direttamente dalla codifica decimale alle istruzioni in assembler. Si potrebbe scrivere un programma "dissamblatore", cioe' un programma che faccia automaticamente questo lavoro. Programmi di questo tipo esistono, non sono molto semplici da scrivere; essi devono essere corredati da una serie di tabelle che per ogni tipo di codice operativo (le istruzioni iniziano tutte con il codice operativo) permettano di risalire ai possibili operandi ed alla lunghezza della istruzione.

Riportiamo un programma che evidenzia al video i contenuti di zone di memoria, dopo aver chiesto all'utente un indirizzo di partenza minore di 4095, dato che il sistema occupa 4K di ROM.

Il programma scrive il contenuto di 16 locazioni partendo dall'indirizzo fornito, e per ogni locazione scrive: l'indirizzo, il valore ricavato con PEEK che e' decimale, il valore esadecimale calcolato e il carattere ottenuto con CHR$. Quest'ultimo carattere puo' interessare quando si indaga sulle tabelle del Basic contenute in ROM.

```
  10 GOTO 1000
 100 PRINT "LISTA SISTEMA OPERATIVO"
 110 PRINT
 120 RETURN
 200 PRINT "BYTE    PEEK    ESADEC. CHR$"
 210 PRINT
 220 FOR K=N TO N+15
 230 LET X=PEEK(K)
 240 LET Y=X/16
 250 LET Z=X-Y*16
 260 LET Y$=CHR$(Y+28)
 270 LET Z$=CHR$(Z+28)
 280 PRINT K,X,Y$;Z$,CHR$(X)
 290 NEXT K
 295 RETURN
1000 GOSUB 100
```

```
1010 PRINT "DA QUALE BYTE ?"
1015 PRINT "SCRIVI 9999 PER USCIRE"
1020 INPUT N
1025 IF N=9999 THEN STOP
1030 IF N<0 OR N>4095 THEN GOTO 1010
1040 CLS
1050 GOSUB 100
1060 GOSUB 200
1070 GOTO 1010
```

Il programma inizia in 1000 e la linea 10 manda a 1000. Seguono, da 100 a 120 il sottoprogramma per il titolo e da 200 a 295 il sottoprogramma per listare la memoria. Il programma chiede da quale byte partire e lista 16 byte. Per uscire dare 9999.

Il sistema operativo puo' essere diviso nelle seguenti parti:

. Dal byte 0 al byte 1873 programmi di gestione tastiera, schermo e registratore.
. Dal byte 1874 al byte 1981 tabella principale dei comandi Basic.
. Dal byte 1982 al byte 3583 interprete Basic.
. Dal byte 3584 al byte 4095 tabella dei caratteri usando 8 byte per ognuno dei 64 caratteri.

Nella prima parte (byte 0-1873) sono contenute le seguenti tabelle:

. Dal byte 108 al byte 185 tabella dei valori corrispondenti ai tasti.
. Dal byte 186 al byte 315 tabella delle parole (estensioni) dei tasti che corrispondono alle parole chiave del Basic. Ogni parola chiave termina con il suo carattere aumentato di 128 per segnalare la fine della parola stessa. I codici ASCII di queste parole vanno da 230 a 254.
. Dal byte 882 al byte 897 tabella che contiene gli indirizzi dei sottoprogrammi di gestione dei movimenti del cursore. I byte sono usati a coppie per contenere questi indirizzi.

Provate a indagare sul contenuto di queste tabelle usando il programma precedente. Quando volete analizzare la tabella che va da 186 a 315 e' meglio se aggiungete al programma la seguente istruzione:

```
235 IF X>127 THEN LET X=X-128
```

per eliminare il 128 sull'ultimo carattere delle parole chiave.

Per ricostruire gli indirizzi contenuti nella terza tabella (882-887), dovete usare la formula: byte-alto * 256 + byte-basso; essi risultano:

Byte	Contenuto		Indirizzo sottopr.	Corrispondenza
882-883	169	3	937	freccia in su
884-885	213	2	725	freccia in giu'
886-887	130	3	898	freccia sin.
888-889	135	3	903	freccia destra
890-891	185	3	953	Home
892-893	203	3	971	Edit
894-895	8	4	1032	NEW-LINE
896-897	149	3	917	Rubout

Altri indirizzi utili dei sottoprogrammi di questa parte del sistema operativo sono:

Indirizzo	Funzione
0	NEW
316	per gestione schermo e tastiera (SCREEN & KEYBOARD)
438	SAVE
518	LOAD
598	LIST
1366	Stampa caratteri aggiungendo 128, cioe' in campo inverso.
1376	Stampa caratteri normali
1474	Fine linea
1627	CLEAR
1697	Stampa numeri
1760	Definizione posizione attuale cursore

1852	Aggiornamento puntatori a riga e colonna video (16421, 16420)
1863	CLS

Inoltre sono indirizzi utili i seguenti:

Byte	Contenuto
647	K in campo inverso usato per il cursore. L in campo inverso si ottiene da questo incrementandolo.
1196	/ usata nel messaggio di errore.
1279	> in campo inverso usato nel puntatore di linea
1312	S in campo inverso, usato per segnalare gli errori e per l'attesa di INPUT.
1706	segno - per i numeri negativi.

Nella parte dell'interprete Basic sono contenute le seguenti tabelle:

. Da 2102 a 2108 tabella associata alla tabella principale dei comandi Basic (1874-1981).

. Da 3008 a 3052 tabella per le funzioni che vengono richiamate usando i comandi scritti carattere per carattere, come PEEK, CHR$, ecc.; questi nomi hanno aggiunto il numero 192 al codice dell'ultimo carattere per segnare la fine della parola. Dopo ogni parola sono disponibili due byte che danno l'indirizzo per il sottoprogramma relativo.

. Da 3359 a 3384 tabella per gli operatori relazionali.

Si segnalano alcuni indirizzi utili di sottoprogrammi contenuti in questa parte:

Indirizzo	Funzione
2122	REM
2339	RANDOMISE
2350	STOP
2365	RUN
2405	RETURN
2417	PRINT
3053	RND
3385	Sottrazione
3390	Addizione
3396	Moltiplicazione
3440	Elevamento a potenza
3472	Divisione
3509	AND
3576	OR

Una routine molto interessante e' quella che inizia a 316 e viene chiamata a 319. Essa gestisce lo schermo e la tastiera. Per mantenere la visione sullo schermo esso deve essere rinfrescato ogni venticinquesimo di secondo, d'altra parte se la configurazione del video cambia troppo rapidamente essa non risulta visibile per l'occhio. Lo ZX80 non rimanda con continuita' fotogrammi al video, ma si interrompe quando svolge altri compiti. La routine in questione svolge le seguenti operazioni:

.1) incrementa il contatore dei fotogrammi;
.2) scandisce la tastiera;
.3) trasferisce sul video il contenuto del display file.

Quando si preme un tasto la routine esce al punto 2); il valore corrispondente al tasto premuto sta nei registri BC. La tastiera e' considerata divisa in 8 parti, considerando i tasti normali usati senza SHIFT; nel registro C sta una configurazione di bit che da' notizia con un bit 0 della parte nella quale e' stato premuto un tasto. Se non e' stato premuto alcun tasto, C contiene tutti bit 1.

Divisione orizzontale tastiera	Contenuto registro C	
. Nessun tasto premuto	11111111	FF
. Parte 0: Z, X, C, V	11111110	FE
. Parte 1: A, S, D, F, G	11111101	FD
. Parte 2: Q, W, E, R, T	11111011	FB
. Parte 3: 1, 2, 3, 4, 5	11110111	F7
. Parte 4: 0, 9, 8, 7, 6	11101111	EF
. Parte 5: P, O, I, U, Y	11011111	DF
. Parte 6: NEW LINE, L, K, J, H	10111111	BF
. Parte 7: spazio, ., M, N, B	01111111	7F

Nel registro B si ha invece notizia della sezione verticale alla quale appartiene il tasto considerando la seguente divisione, tenendo anche conto dello SHIFT.

Divisione verticale	Reg. B		Reg. B con SHIFT	
.0) Nessun tasto	11111111	FF	11111110	FE
.1) Spazio, NEW LINE P, 0, 1, Q, A	11111101	FD	11111100	FC
.2) ., L, O, 9, 2, W, S, Z	11111011	FB	11111010	FA
.3) M, K, I, 8, 3, E, D, X	11110111	F7	11110110	F6
.4) N, J, U, 7, 4, R, F, C	11101111	EF	11101110	EE
.5) B, H, Y, 6, 5, T, G, V	11011111	DF	11011110	DE

Provate il programma che segue, il quale pone l'indirizzo della routine di cui sopra (entrata 319 decimale e quindi

013F in esadecimale) in 30000, poi carica le istruzioni per trasferire il contenuto dei registri BC in HL. Date il RUN a questo programma, poi scrivete in modo immediato PRINT USR(30000) e subito dopo il NEW LINE premete un qualunque tasto. Vedrete apparire nell'angolo in alto a sinistra del video il contenuto di HL e quindi di BC in decimale.

```
10 POKE 30000,205        (CD        CALL)
20 POKE 30001,63         (3F        3F)
30 POKE 30002,1          (01        01)
40 POKE 30003,96         (60        LD H,B)
50 POKE 30004,105        (69        LD L,C)
60 POKE 30005,201        (C9        RET)
```

Quando premete NEW LINE dopo RUN il comando resta nella parte bassa del video ed il cursore segna L, premete il tasto voluto senza NEW LINE dopo.

Se premete 2 vedete apparire -1033, che corrisponde in esadecimale a FBF7, cioe' il valore di B seguito dal valore di C, ma attenzione al calcolo:

```
   F      B      F      7
 1111   1101   1111   0111
                               numero negativo
                               complementato
 1111   1011   1111   0110     tolgo 1
 0000   0100   0000   1001     scambio 0 con 1
   0      4      0      9      valore assoluto
               1033            in decimale.
```

Il sistema sfrutta questa situazione dei registri BC per andare a ricercare nelle tabelle il codice del carattere.

Il display file viene ingrandito mentre il video si riempie. Alla partenza del programma Basic viene messo un NEW LINE nella prima posizione (indirizzo contenuto in D-FILE). Se il programma scrive qualcosa sul video il display file si ingrandisce; una PRINT a vuoto fa aggiungere un NEW LINE. PRINT "AB" fa aggiungere i due caratteri AB ed un NEW LINE. Il display file deve essere completato quando per una qualunque ragione il sistema deve fare apparire il cursore. Le ragioni possono essere:

. esecuzione ultima linea del programma;
. STOP in programma;
. richiesta di INPUT;
. lo schermo e' pieno;
. manca memoria;
. segnalazione di errore.

In questi casi il sistema completa il display file lavorando sui byte (16420, 16421) che danno la posizione corrente sul video.

IL SISTEMA OPERATIVO DELLO ZX81 E ZX80-NUOVA ROM

Si riporta il listato della parte del Sistema Operativo che si trova memorizzato in ROM dall'indirizzo 0 all'indirizzo 119 decimale. Nel listato si riportano gli indirizzi dei byte in esadecimale, il codice macchina in esadecimale e le istruzioni in Assembler.

Da 120 a 203 si trova la tabella dei caratteri. Da 204 a 242 si trova la tabella dei tasti usati in stato F. Da 243 a 272 si trova la tabella dei tasti usati in stato G. Da 273 a 507 si trova la tabella della estensione delle parole chiave memorizzate con l'ultimo carattere invertito (+128).

Ind.	Codice	Assembler	Ind.	Codice	Assembler
0000	D3 FD	OUT (FD),A	0035	C3 88 14	JP 1488
0002	01 FF 7F	LD BC,7FFF	0038	0D	DEC C
0005	C3 CB 03	JP 03CB	0039	C2 45 00	JP NZ,0045
0008	2A 16 40	LD HL,(4016)	003C	E1	POP HL
000B	22 18 40	LD (4018),HL	003D	05	DEC B
000E	18 46	JR 0056	003E	C8	RET Z
0010	A7	AND A	003F	CB D9	SET 3,C
0011	C2 F1 07	JP NZ,07F1	0041	ED 4F	LD R,A
0014	C3 F5 07	JP 07 F5	0043	FB	EI
0017	FF	RST 38	0044	E9	JP (HL)
0018	2A 16 40	LD HL,(4016)	0045	D1	POP DE
001B	7E	LD A,(HL)	0046	C8	RET Z
001C	A7	AND A	0047	18 F8	JR 0041
001D	C0	RET NZ	0049	2A 16 40	LD HL,(4016)
001E	00	NOP	004C	23	INC HL
001F	00	NOP	004D	22 16 40	LD (4016),HL
0020	CD 49 00	CALL 0049	0050	7E	LD A,(HL)
0023	18 F7	JR 001C	0051	FE 7F	CP 7F
0025	FF	RST 38	0053	C0	RET NZ
0026	FF	RST 38	0054	18 F6	JR 004C
0027	FF	RST 38	0056	E1	POP HL
0028	C3 9D 19	JP 199D	0057	6E	LD L,(HL)
002B	F1	POP AF	0058	FD 75 00	LD (IY),L
002C	D9	EXX	005B	ED 7B 02 40	LD SP,(4002)
002D	E3	EX (SP),HL	005F	CD 07 02	CALL 0207
002E	D9	EXX	0062	C3 BC 14	JP 14BC
002F	C9	RET	0065	FF	RST 38
0030	C5	PUSH BC	0066	08	EX AF,AF'
0031	2A 14 40	LD HL,(4014)	0067	3C	INC A
0034	E5	PUSH HL	0068	FA 6D 00	JP M,006D

Ind.	Codice	Assembler	Ind.	Codice	Assembler
006B	28 02	JR Z,006F	0073	E5	PUSH HL
006D	08	EX AF,AF'	0074	2A 0C 40	LD HL,(400C)
006E	C9	RET	0077	CB FC	SET 7,H
006F	08	EX AF,AF'	0079	76	HALT
0070	F5	PUSH AF	007A	D3 FD	OUT (FD),A
0071	C5	PUSH BC	007C	DD E9	JP (IX)
0072	D5	PUSH DE			

Programma per listare le tabelle. Il programma chiede l'indirizzo di inizio e l'indirizzo di fine zona da listare. Si ha la stampa di 16 righe, poi una pausa, che puo' essere interrotta dalla pressione di un qualunque tasto, e dopo la pulizia del video, vengono evidenziate altre 16 righe.

```
  10 GOTO 1000
 100 PRINT "LISTA TABELLA"
 110 PRINT
 120 RETURN
 200 PRINT "BYTE   ESADEC.   CHR$"
 210 PRINT
 220 FOR K= N TO M STEP 16
 230 FOR J= 0 TO 15
 235 IF(K+J)>M THEN RETURN
 240 LET X=PEEK(K+J)
 250 LET H=INT(X/16)
 260 LET L=X-H*16
 270 PRINT K+J;TAB 7;CHR$(H+28);CHR$(L+28);TAB(17);CHR$ X
 280 NEXT J
 290 PAUSE 4000
 300 NEXT K
 310 RETURN
1000 CLS
1010 GOSUB 100
1020 PRINT "INDIRIZZO INIZIO: ";
1030 INPUT N
1035 PRINT N
1040 PRINT "INDIRIZZO FINE: ";
1050 INPUT M
1055 PRINT M
1060 IF N>0 AND M>0 AND M<8191 AND M>N THEN GOTO 1070
1065 GOTO 1000
1070 CLS
1075 GOSUB 100
1080 GOSUB 200
1090 STOP
```

Si segnalano alcuni indirizzi di particolare interesse situati nella prima parte del sistema operativo gia' listata.

. 0008 e' il punto di entrata della routine per il trattamento dell'errore. Viene chiamata con RST 0008, e dopo ci deve essere un byte con il codice dell'errore.

```
   Esempio:        30000   RST 0008   CF
                   30001   NOP        0D
```

tratta l'errore D e quindi 13.

. 0010 e' il punto di entrata della routine per stampare un carattere. Viene chiamata con RST 0010. Prima di chiamare questa routine si deve porre nell'accumulatore il codice del carattere da stampare. RST 0010 corrisponde al codice D7.

. 0018 e' il punto di entrata di una routine per raggiungere il carattere successivo in una linea di programma Basic. Si chiama con RST 0018 corrispondente al codice DF.

. 0020 e' il punto di entrata di un'altra routine simile alla precedente. Si chiama con RST 0020 corrispondente al codice E7.

. 0028 e' il punto di entrata per la routine che svolge i calcoli dei numeri in forma esponenziale. Tale routine e' situata a partire dall'indirizzo 199C. Si chiama con RST 0028 corrispondente al codice EF.

. 0030 e' il punto di entrata della routine che predispone un'area di BC spazi nella zona delle variabili. Si chiama con RST 0030 corrispondente al codice F7.

. 0038 e' il punto di entrata della routine di servizio degli interrupt che gestiscono le linee sullo schermo. Si chiama con RST 0038 corrispondente al codice FF.

. 0066 e' il punto di entrata della routine che serve NMI (interrupt non mascherabile) e manda fotogrammi al video dopo un interrupt non mascherabile quando il calcolatore funziona in modo SLOW.

Nelle pagine seguenti si riporta la lista del Sistema Operativo dall'indirizzo 508 all'indirizzo 3112 decimale.

Dall'indirizzo 3113 (0C29 esadecimale) all'indirizzo 3257 (0CB9 esadecimale) si trova la tabella della sintassi del linguaggio. In essa una prima parte e' costituita dai puntatori alla seconda parte e per ogni comando si trovano gli indirizzi delle relative routine. Tale tabella puo' essere listata con il programma precedentemente riportato. In quel programma si possono sostituire i comandi di PRINT delle tabelle con dei comandi LPRINT se si dispone della stampante.

Ind.	Codice	Assembler
01FC	23	INC HL
01FD	EB	EX DE,HL
01FE	2A 14 40	LD HL,(4014)
0201	37	SCF
0202	ED 52	SBC HL, DE
0204	EB	EX DE,HL
0205	D0	RET NC
0206	E1	POP HL
0207	21 3B 40	LD HL,403B
020A	7E	LD A,(HL)
020B	17	RLA
020C	AE	XOR (HL)
020D	17	RLA
020E	D0	RET NC
020F	3E 7F	LD A,7F
0211	08	EX AF,AF'
0212	06 11	LD B,11
0214	D3 FE	OUT (FE),A
0216	10 FE	DJNZ 0216
0218	D3 FD	OUT (FD),A
021A	08	EX AF,AF'
021B	17	RLA
021C	30 08	JR NC,0226
021E	CB FE	SET 7,(HL)
0220	F5	PUSH AF
0221	C5	PUSH BC
0222	D5	PUSH DE
0223	E5	PUSH HL
0224	18 03	JR 0229
0226	CB B6	RES 6,(HL)
0228	C9	RET
0229	2A 34 40	LD HL,(4034)
022C	2B	DEC HL
022D	3E 7F	LD A,7F
022F	A4	AND H
0230	B5	OR L
0231	7C	LD A,H
0232	20 03	JR NZ,0237
0234	17	RLA
0235	18 02	JR 0239
0237	46	LD B,(HL)
0238	37	SCF
0239	67	LD H,A
023A	22 34 40	LD (4034),HL
023D	D0	RET NC
023E	CD BB 02	CALL 02BB
0241	ED 4B 25 40	LD BC,(4025)
0245	22 25 40	LD (4025),HL
0248	78	LD A,B
0249	C6 02	ADD A,02
024B	ED 42	SBC HL,BC
024D	3A 27 40	LD A,(4027)
0250	B4	DRH
0251	B5	DRL
0252	58	LD E,B
0253	06 0B	LD B,0B
0255	21 3B 40	LD HL,403B
0258	CB 86	RES 0,(HL)
025A	20 08	JR NZ,0264
025C	CB 7E	BIT 7,(HL)
025E	CB C6	SET 0,(HL)
0260	C8	RET Z
0261	05	DEC B
0262	00	NOP
0263	37	SCF
0264	21 27 40	LD HL, 4027
0267	3F	CCF
0268	CB 10	RLB
026A	10 FE	DJNZ 026A
026C	46	LD B,(HL)
026D	7B	LD A,E
026E	FE FE	CP FE
0270	9F	SBC A,A
0271	06 1F	LD B,1F
0273	B6	OR (HL)
0274	A0	AND B
0275	1F	RRA
0276	77	LD (HL),A
0277	D3 FF	OUT (FF),A
0279	2A 0C 40	LD HL,(400C)
027C	CB FC	SET 7,H
027E	CD 92 02	CALL 0292
0281	ED 5F	LD A,R
0283	01 01 19	LD BC,1901
0286	3E F5	LD A,F5
0288	CD B5 02	CALL 02B5
028B	2B	DEC HL
028C	CD 92 02	CALL 0292
028F	C3 29 02	JP 0229
0292	DD E1	POP IX
0294	FD 4E 28	LD C,(IY+28)
0297	FD CB 3B 7E	BIT 7,(IY+3B)
029B	28 0C	JR Z,02A9
029D	79	LD A,C
029E	ED 44	NEG
02A0	3C	INC A
02A1	08	EX AF,AF'
02A2	D3 FE	OUT (FE),A
02A4	E1	POP HL
02A5	D1	POP DE

Ind.	Codice	Assembler
02A6	C1	POP BC
02A7	F1	POP AF
02AB	C9	RET
02A9	3E FC	LD A,FC
02AB	06 01	LD B,01
02AD	CD B5 02	CALL 02B5
02B0	2B	DEC HL
02B1	E3	EX (SP),HL
02B2	E3	EX (SP),HL
02B3	DD E9	JP (IX)
02B5	ED 4F	LD R,A
02B7	3E DD	LD A,DD
02B9	FB	EI
02BA	E9	JP (HL)
02BB	21 FF FF	LD HL,FFFF
02BE	01 FE FE	LD BC,FEFE
02C1	ED 78	IN A,(C)
02C3	F6 01	OR 01
02C5	F6 E0	OR E0
02C7	57	LD D,A
02C8	2F	CPL
02C9	FE 01	CP 01
02CB	9F	SBC A,A
02CC	B0	OR B
02CD	A5	AND L
02CE	6F	LD L,A
02CF	7C	LD A,H
02D0	A2	AND D
02D1	67	LD H,A
02D2	CB 00	RLC B
02D4	ED 78	IN A,(C)
02D6	38 ED	JR C,02C5
02D8	1F	RRA
02D9	CB 14	RL H
02DB	17	RLA
02DC	17	RLA
02DD	17	RLA
02DE	9F	SBC A,A
02DF	E6 18	AND 18
02E1	C6 1F	ADD A,1F
02E3	32 28 40	LD (4028),A
02E6	C9	RET
02E7	FD CB 3B 7E	BIT 7,(IY+3B)
02EB	C8	RET Z
02EC	76	HALT
02ED	D3 FD	OUT (FD),A
02EF	FD CB 3B BE	RES 7,(IY+3B)
02F3	C9	RET
02F4	CF	RST 8
02F5	0E CD	LD C,CD

Ind.	Codice	Assembler
02F7	A8	XOR B
02F8	03	INC BC
02F9	38 F9	JR C,02F4
02FB	EB	EX DE,HL
02FC	11 CB 12	LD DE,12CB
02FF	CD 46 0F	CALL 0F46
0302	30 2E	JR NC,0332
0304	10 FE	DJNZ 0304
0306	1B	DEC DE
0307	7A	LD A,D
0308	B3	OR E
0309	20 F4	JR NZ,02FF
030B	CD 1E 03	CALL 031E
030E	CB 7E	BIT 7,(HL)
0310	23	INC HL
0311	28 F8	JR Z,030B
0313	21 09 40	LD HL,4009
0316	CD 1E 03	CALL 031E
0319	CD FC 01	CALL 01FC
031C	18 F8	JR 0316
031E	5E	LD E,(HL)
031F	37	SCF
0320	CB 13	RL E
0322	C8	RET Z
0323	9F	SBC A,A
0324	E6 05	AND 05
0326	C6 04	ADD A,04
0328	4F	LD C,A
0329	D3 FF	OUT (FF),A
032B	06 23	LD B,23
032D	10 FE	DJNZ 032D
032F	CD 46 0F	CALL 0F46
0332	30 72	JR NC,03A6
0334	06 1E	LD B,1E
0336	10 FE	DJNZ 0336
0338	0D	DEC C
0339	20 EE	JR NZ,0329
033B	A7	AND A
033C	10 FD	DJNZ 033B
033E	18 E0	JR 0320
0340	CD A8 03	CALL 03A8
0343	CB 12	RL D
0345	CB 0A	RRC D
0347	CD 4C 03	CALL 034C
034A	18 FB	JR 0347
034C	0E 01	LD C,01
034E	06 00	LD B,00
0350	3E 7F	LD A,7F
0352	DB FE	IN A,(FE)
0354	D3 FF	OUT (FF),A

Ind.	Codice	Assembler
0356	1F	RRA
0357	30 49	JR NC,03A2
0359	17	RLA
035A	17	RLA
035B	38 28	JR C,0385
035D	10 F1	DJNZ 0350
035F	F1	POP AF
0360	BA	CP D
0361	D2 E5 03	JP NC,03E5
0364	62	LD H,D
0365	6B	LD L,E
0366	CD 4C 03	CALL 034C
0369	CB 7A	BIT 7,D
036B	79	LD A,C
036C	20 03	JR NZ,0371
036E	BE	CP (HL)
036F	20 D6	JR NZ,0347
0371	23	INC HL
0372	17	RLA
0373	30 F1	JR NC,0366
0375	FD 34 15	INC (IY+15)
0378	21 09 40	LD HL,4009
037B	50	LD D,B
037C	CD 4C 03	CALL 034C
037F	71	LD (HL),C
0380	CD FC 01	CALL 01FC
0383	18 F6	JR 037B
0385	D5	PUSH DE
0386	1E 94	LD E,94
0388	06 1A	LD B,1A
038A	1D	DEC E
038B	DB FE	IN A,(FE)
038D	17	RLA
038E	CB 7B	BIT 7,E
0390	7B	LD A,E
0391	38 F5	JR C,0388
0393	10 F5	DJNZ 038A
0395	D1	POP DE
0396	20 04	JR NZ,039C
0398	FE 56	CP 56
039A	30 B2	JR NC,034E
039C	3F	CCF
039D	CB 11	RL C
039F	30 AD	JR NC,034E
03A1	C9	RET
03A2	7A	LD A,D
03A3	A7	AND A
03A4	28 BB	JR Z,0361
03A6	CF	RST 8
03A7	0C	INC C
03A8	CD 55 0F	CALL 0F55
03AB	3A 01 40	LD A,(4001)
03AE	87	ADD A,A
03AF	FA 9A 0D	JP N,0D9A
03B2	E1	POP HL
03B3	D0	RET NC
03B4	E5	PUSH HL
03B5	CD E7 02	CALL 02E7
03B8	CD F8 13	CALL 13F8
03BB	62	LD H,D
03BC	6B	LD L,E
03BD	0D	DEC C
03BE	F8	RET M
03BF	09	ADD HL,BC
03C0	CB FE	SET 7,(HL)
03C2	C9	RET
03C3	CD E7 02	CALL 02E7
03C6	ED 4B 04 40	LD BC,(4004)
03CA	0B	DEC BC
03CB	60	LD H,B
03CC	69	LD L,C
03CD	3E 3F	LD A,3F
03CF	36 02	LD (HL),02
03D1	2B	DEC HL
03D2	BC	CP H
03D3	20 FA	JR NZ,03CF
03D5	A7	AND A
03D6	ED 42	SBC HL,BC
03D8	09	ADD HL,BC
03D9	23	INC HL
03DA	30 06	JR NC,03E2
03DC	35	DEC (HL)
03DD	28 03	JR Z,03E2
03DF	35	DEC (HL)
03E0	28 F3	JR Z,03D5
03E2	22 04 40	LD (4004),HL
03E5	2A 04 40	LD HL,(4004)
03E8	2B	DEC HL
03E9	36 3E	LD (HL),3E
03EB	2B	DEC HL
03EC	F9	LD SP,HL
03ED	2B	DEC HL
03EE	2B	DEC HL
03EF	22 02 40	LD (4002),HL
03F2	3E 1E	LD A,1E
03F4	ED 47	LD I,A
03F6	ED 56	IM1
03F8	FD 21 00 40	LD IY,4000
03FC	FD 36 3B 40	LD (IY+3D),40
0400	21 7D 40	LD HL,407D

Ind.	Codice	Assembler
0403	22 0C 40	LD (400C),HL
0406	06 19	LD B,19
0408	36 76	LD (HL),76
040A	23	INC HL
040B	10 FB	DJNZ 0408
040D	22 10 40	LD (4010),HL
0410	CD 9A 14	CALL 149A
0413	CD AD 14	CALL 14AD
0416	CD 07 02	CALL 0207
0419	CD 2A 0A	CALL 0A2A
041C	2A 0A 40	LD HL,(400A)
041F	ED 5B 23 40	LD DE,(4023)
0423	A7	AND A
0424	ED 52	SBC HL,DE
0426	EB	EX DE,HL
0427	30 04	JR NC,042D
0429	19	ADD HL,DE
042A	22 23 40	LD (4023),HL
042D	CD D8 09	CALL 09D8
0430	28 01	JR Z,0433
0432	EB	EX DE,HL
0433	CD 3E 07	CALL 073E
0436	FD 35 1E	DEC (IY+1E)
0439	20 37	JR NZ,0472
043B	2A 0A 40	LD HL,(400A)
043E	CD D8 09	CALL 09D8
0441	2A 16 40	LD HL,(4016)
0444	37	SCF
0445	ED 52	SBC HL,DE
0447	21 23 40	LD HL,4023
044A	30 0B	JR NC,0457
044C	EB	EX DE,HL
044D	7E	LD A,(HL)
044E	23	INC HL
044F	ED A0	LDI
0451	12	LD (DE),A
0452	18 C5	JR 0419
0454	21 0A 40	LD HL,400A
0457	5E	LD E,(HL)
0458	23	INC HL
0459	56	LD D,(HL)
045A	E5	PUSH HL
045B	EB	EX DE,HL
045C	23	INC HL
045D	CD D8 09	CALL 09D8
0460	CD BB 05	CALL 05BB
0463	E1	POP HL
0464	FD CB 2D 6E	BIT 5,(IY+2D)
0468	20 08	JR NZ,0472
046A	72	LD (HL),D
046B	2B	DEC HL
046C	73	LD (HL),E
046D	18 AA	JR 0419
046F	CD AD 14	CALL 14AD
0472	2A 14 40	LD HL,(4014)
0475	7E	LD A,(HL)
0476	FE 7E	CP 7E
0478	20 08	JR NZ,0482
047A	01 06 00	LD BC,0006
047D	CD 60 0A	CALL 0A60
0480	18 F3	JR 0475
0482	FE 76	CP 76
0484	23	INC HL
0485	20 EE	JR NZ,0475
0487	CD 37 05	CALL 0537
048A	CD 1F 0A	CALL 0A1F
048D	2A 14 40	LD HL,(4014)
0490	FD 36 00 FF	LD (IY),FF
0494	CD 66 07	CALL 0766
0497	FD CB 00 7E	BIT 7,(IY)
049B	20 24	JR NZ,04C1
049D	3A 22 40	LD A,(4022)
04A0	FE 18	CP 18
04A2	30 1D	JR NC,04C1
04A4	3C	INC A
04A5	32 22 40	LD (4022),A
04A8	47	LD B,A
04A9	0E 01	LD C,01
04AB	CD 18 09	CALL 0918
04AE	54	LD D,H
04AF	5D	LD E,L
04B0	7E	LD A,(HL)
04B1	2B	DEC HL
04B2	BE	CP (HL)
04B3	20 FC	JR NZ,04B1
04B5	23	INC HL
04B6	EB	EX DE,HL
04B7	3A 05 40	LD A,(4005)
04BA	FE 4D	CP 4D
04BC	DC 5D 0A	CALL C,0A5D
04BF	18 C9	JR 048A
04C1	21 00 00	LD HL,0000
04C4	22 18 40	LD (4018),HL
04C7	21 3B 40	LD HL,403B
04CA	CB 7E	BIT 7,(HL)
04CC	CC 29 02	CALL Z,0229
04CF	CB 46	BIT 0,(HL)
04D1	28 FC	JR Z,04CF
04D3	ED 4B 25 40	LD BC,(4025)
04D7	CD 4B 0F	CALL 0F4B

Ind.	Codice	Assembler	Ind.	Codice	Assembler
04DA	CD BD 07	CALL 07BD	053A	FD CB 2D 6E	BIT 5,(IY+2D)
04DD	30 93	JR NC,0472	053E	20 16	JR NZ,0556
04DF	3A 0640	LD A,(4006)	0540	FD CB 01 96	RES 2,(IY+01)
04E2	3D	DEC A	0544	7E	LD A,(HL)
04E3	FA 08 05	JP M,0508	0545	FE 7F	CP 7F
04E6	20 0F	JR NZ,04F7	0547	C8	RET Z
04E8	32 06 40	LD (4006),A	0548	23	INC HL
04EB	1D	DEC E	0549	CD B4 07	CALL 07B4
04EC	7B	LD A,E	054C	28 F6	JR Z,0544
04ED	D6 27	SUB 27	054E	FE 26	CP 26
04EF	38 01	JR C,04F2	0550	38 F2	JR C,0544
04F1	5F	LD E,A	0552	FE DE	CP DE
04F2	21 CC 00	LD HL,00CC	0554	28 EA	JR Z,0540
04F5	18 0E	JR 05 05	0556	FD CB 01 D6	SET 2,(IY+01)
04F7	7E	LD A,(HL)	055A	18 E8	JR 0544
04F8	FE 76	CP 76	055C	01 01 00	LD BC,0001
04FA	28 2F	JR Z,052B	055F	C3 60 0A	JP 0A60
04FC	FE 40	CP 40	0562	9F	SBC A,A
04FE	CB FF	SET 7,A	0563	05	DEC B
0500	38 19	JR C,051B	0564	54	LD D,H
0502	21 C7 00	LD HL,00C7	0565	04	INC B
0505	19	ADD HL,DE	0566	76	HALT
0506	18 0D	JR 05 15	0567	05	DEC B
0508	7E	LD A,(HL)	0568	7F	LD A,A
0509	FD CB 01 56	BIT 2,(IY+01)	0569	05	DEC B
050D	20 07	JR NZ,0516	056A	AF	XOR A
050F	C6 C0	ADD A,C0	056B	05	DEC B
0511	FE E6	CP E6	056C	C4 05 0C	CALL NZ,0C05
0513	30 01	JR NC,0516	056F	06 8B	LD B,8B
0515	7E	LD A,(HL)	0571	05	DEC B
0516	FE F0	CP F0	0572	AF	XOR A
0518	EA 2D 05	JP PE,052D	0573	05	DEC B
051B	5F	LD E,A	0574	AF	XOR A
051C	CD 37 05	CALL 0537	0575	05	DEC B
051F	7B	LD A,E	0576	CD 93 05	CALL 0593
0520	CD 26 05	CALL 0526	0579	7E	LD A,(HL)
0523	C3 72 04	JP 0472	057A	36 7F	LD (HL),7F
0526	CD 9B 09	CALL 099B	057C	23	INC HL
0529	12	LD (DE),A	057D	18 09	JR 0588
052A	C9	RET	057F	23	INC HL
052B	3E 78	LD A,78	0580	7E	LD A,(HL)
052D	5F	LD E,A	0581	FE 76	CP 76
052E	21 82 04	LD HL,0482	0583	28 18	JR Z,059D
0531	19	ADD HL,DE	0585	36 7F	LD (HL),7F
0532	19	ADD HL,DE	0587	2B	DEC HL
0533	4E	LD C,(HL)	0588	77	LD (HL),A
0534	23	INC HL	0589	18 98	JR 0523
0535	46	LD B,(HL)	058B	CD 93 05	CALL 0593
0536	C5	PUSH BC	058E	CD 5C 05	CALL 055C
0537	2A 14 40	LD HL,(4014)	0591	18 F6	JR 0589

Ind.	Codice	Assembler
0593	2B	DEC HL
0594	ED 5B 14 40	LD DE,(4014)
0598	1A	LD A,(DE)
0599	FE 7F	CP 7F
059B	C0	RET NZ
059C	D1	POP DE
059D	18 EA	JR 0589
059F	2A 0A 40	LD HL,(400A)
05A2	CD D8 09	CALL 09D8
05A5	EB	EX DE,HL
05A6	CD BB 05	CALL 05BB
05A9	21 0B 40	LD HL,400B
05AC	C3 64 04	JP 0464
05AF	7B	LD A,E
05B0	E6 07	AND 07
05B2	32 06 40	LD (4006),A
05B5	18 E6	JR 059D
05B7	EB	EX DE,HL
05B8	11 C2 04	LD DE,04C2
05BB	7E	LD A,(HL)
05BC	E6 C0	AND C0
05BE	20 F7	JR NZ,05B7
05C0	56	LD D,(HL)
05C1	23	INC HL
05C2	5E	LD E,(HL)
05C3	C9	RET
05C4	CD 1F 0A	CALL 0A1F
05C7	21 6F 04	LD HL,046F
05CA	E5	PUSH HL
05CB	FD CB 2D 6E	BIT 5,(IY+2D)
05CF	C0	RET NZ
05D0	2A 14 40	LD HL,(4014)
05D3	22 0E 40	LD (400E),HL
05D6	21 21 18	LD HL,1821
05D9	22 39 40	LD (4039),HL
05DC	2A 0A 40	LD HL,(400A)
05DF	CD D8 09	CALL 09D8
05E2	CD BB 05	CALL 05BB
05E5	7A	LD A,B
05E6	B3	OR E
05E7	C8	RET Z
05E8	2B	DEC HL
05E9	CD A5 0A	CALL 0AA5
05EC	23	INC HL
05ED	4E	LD C,(HL)
05EE	23	INC HL
05EF	46	LD B,(HL)
05F0	23	INC HL
05F1	ED 5B 0E 40	LD DE,(400E)
05F5	3E 7F	LD A,7F

Ind.	Codice	Assembler
05F7	12	LD(DE),A
05F8	13	INC DE
05F9	E5	PUSH HL
05FA	21 1D 00	LD HL,001D
05FD	19	ADD HL,DE
05FE	09	ADD HL,BC
05FF	ED 72	SBC HL,SP
0601	E1	POP HL
0602	D0	RET NC
0603	ED B0	LD IR
0605	EB	EX DE,HL
0606	D1	POP DE
0607	CD A6 14	CALL 14A6
060A	18 91	JR 059D
060C	CD 1F 0A	CALL 0A1F
060F	21 72 04	LD HL,0472
0612	FD CB 2D 6E	BIT 5,(IY+2D)
0616	20 11	JR NZ,0629
0618	2A 14 40	LD HL,(4014)
061B	7E	LD A,(HL)
061C	FE FF	CP FF
061E	28 06	JR Z,0626
0620	CD E2 08	CALL 08E2
0623	CD 2A 0A	CALL 0A2A
0626	21 19 04	LD HL,0419
0629	E5	PUSH HL
062A	CD BA 0C	CALL 0CBA
062D	E1	POP HL
062E	CD 37 05	CALL 0537
0631	CD 5C 05	CALL 055C
0634	CD 73 0A	CALL 0A73
0637	20 15	JR NZ,064E
0639	78	LD A,B
063A	B1	OR C
063B	C2 E0 06	JP NZ,06E0
063E	0B	DEC BC
063F	0B	DEC BC
0640	ED 43 07 40	LD (4007),BC
0644	FD 36 22 02	LD (IY+2),02
0648	ED 5B 0C 40	LD DE,(400C)
064C	18 13	JR 0661
064E	FE 76	CP 76
0650	28 12	JR Z,0664
0652	ED 4B 30 40	LD BC,(4030)
0656	CD 18 09	CALL 0918
0659	ED 5B 29 40	LD DE,(4029)
065D	FD 36 22 02	LD (IY+22),02
0661	DF	RST 18
0662	FE 76	CP 76
0664	CA 13 04	JP Z,0413

Ind.	Codice	Assembler	Ind.	Codice	Assembler
0667	FD 36 01 80	LD (IY+01),80	06D7	CD 98 0A	CALL 0A98
066B	EB	EX DE,HL	06DA	CD AD 14	CALL 14AD
066C	22 29 40	LD (4029),HL	06DD	C3 C1 04	JP 04C1
066F	EB	EX DE,HL	06E0	ED 43 0A 40	LD (400A),BC
0670	CD 4D 00	CALL 004D	06E4	2A 16 40	LD HL,(4016)
0673	CD C1 0C	CALL 0CC1	06E7	EB	EX DE,HL
0676	FD CB 01 8E	RES 1,(IY+01)	06E8	21 13 04	LD HL,0413
067A	3E C0	LD A,C0	06EB	E5	PUSH HL
067C	FD 77 19	LD (IY+19),A	06EC	2A 1A 40	LD HL,(401A)
067F	CD A3 14	CALL 14A3	06EF	ED 52	SBC HL,DE
0682	FD CB 2D AE	RES 5,(IY+2D)	06F1	E5	PUSH HL
0686	FD CB 00 7E	BIT 7,(IY)	06F2	C5	PUSH BC
068A	28 22	JR Z,06AE	06F3	CD E7 02	CALL 02E7
068C	2A 29 40	LD HL,(4029)	06F6	CD 2A 0A	CALL 0A2A
068F	A6	AND (HL)	06F9	E1	POP HL
0690	20 1C	JR NZ,06AE	06FA	CD D8 09	CALL 09D8
0692	56	LD D,(HL)	06FD	20 06	JR NZ,0705
0693	23	INC HL	06FF	CD F2 09	CALL 09F2
0694	5E	LD E,(HL)	0702	CD 60 0A	CALL 0A60
0695	ED 53 07 40	LD (4007),DE	0705	C1	POP BC
0699	23	INC HL	0706	79	LD A,C
069A	5E	LD E,HL	0707	3D	DEC A
069B	23	INC HL	0708	B0	OR B
069C	56	LD D,(HL)	0709	C8	RET Z
069D	23	INC HL	070A	C5	PUSH BC
069E	EB	EX DE,HL	070B	03	INC BC
069F	19	ADD HL,DE	070C	03	INC BC
06A0	CD 46 0F	CALL 0F46	070D	03	INC BC
06A3	38 C7	JR C,066C	070E	03	INC BC
06A5	21 00 40	LD HL,4000	070F	2B	DEC HL
06A8	CB 7E	BIT 7,(HL)	0710	CD 9E 09	CALL 099E
06AA	28 02	JR Z,06AE	0713	CD 07 02	CALL 0207
06AC	36 0C	LD (HL),0C	0716	C1	POP BC
06AE	FD CB 38 7E	BIT 7,(IY+38)	0717	C5	PUSH BC
06B2	CC 71 08	CALL Z,0871	0718	13	INC DE
06B5	01 21 01	LD BC,0121	0719	2A 1A 40	LD HL,(401A)
06B8	CD 18 09	CALL 0918	071C	2B	DEC HL
06BB	3A 00 40	LD A,(4000)	071D	ED B8	LD DR
06BE	ED 4B 07 40	LD BC,(4007)	071F	2A 0A 40	LD HL,(400A)
06C2	3C	INC A	0722	EB	EX DE,HL
06C3	28 0C	JR Z,06D1	0723	C1	POP BC
06C5	FE 09	CP 09	0724	70	LD (HL),B
06C7	20 01	JR NZ,06CA	0725	2B	DEC HL
06C9	03	INC BC	0726	71	LD (HL),C
06CA	ED 43 2B 40	LD (402B),BC	0727	2B	DEC HL
06CE	20 01	JR NZ,06D1	0728	73	LD (HL),E
06D0	0B	DEC BC	0729	2B	DEC HL
06D1	CD EB 07	CALL 07EB	072A	72	LD (HL),D
06D4	3E 18	LD A,18	072B	C9	RET
06D6	D7	RST 10	072C	FD CB 01 CE	SET 1,(IY+01)

Ind.	Codice	Assembler	Ind.	Codice	Assembler
0730	CD A7 0E	CALL 0EA7	0798	18 D3	JR 076D
0733	78	LD A,B	079A	D7	RST 10
0734	E6 3F	AND 3F	079B	18 D0	JR 076D
0736	67	LD H,A	079D	3A 06 40	LD A,(4006)
0737	69	LD L,C	07A0	06 AB	LD B,AB
0738	22 0A 40	LD (400A),HL	07A2	A7	AND A
073B	CD D8 09	CALL 09D8	07A3	20 05	JR NZ,07AA
073E	1E 00	LD E,00	07A5	3A 01 40	LD A,(4001)
0740	CD 45 07	CALL 0745	07A8	06 B0	LD B,B0
0743	18 FB	JR 0740	07AA	1F	RRA
0745	ED 4B 0A 40	LD BC,(400A)	07AB	1F	RRA
0749	CD EA 09	CALL 09EA	07AC	E6 01	AND 01
074C	16 92	LD D,92	07AE	80	ADD A,B
074E	28 05	JR Z,0755	07AF	CD F5 07	CALL 07F5
0750	11 00 00	LD DE,0000	07B2	18 B9	JR 076D
0753	CB 13	RL E	07B4	FE 7E	CP 7E
0755	FD 73 1E	LD (IY+1E),E	07B6	C0	RET NZ
0758	7E	LD A,(HL)	07B7	23	INC HL
0759	FE 40	CP 40	07B8	23	INC HL
075B	C1	POP BC	07B9	23	INC HL
075C	D0	RET NC	07BA	23	INC HL
075D	C5	PUSH BC	07BB	23	INC HL
075E	CD A5 0A	CALL 0AA5	07BC	C9	RET
0761	23	INC HL	07BD	16 00	LD D,00
0762	7A	LD A,D	07BF	CB 28	SRA B
0763	D7	RST 10	07C1	9F	SBC A,A
0764	23	INC HL	07C2	F6 26	OR 26
0765	23	INC HL	07C4	2E 05	LD L,05
0766	22 16 40	LD (4016),HL	07C6	95	SUB L
0769	FD CB 01 C6	SET 0,(IY+01)	07C7	85	ADD A,L
076D	ED 4B 18 40	LD BC,(4018)	07C8	37	SCF
0771	2A 16 40	LD HL,(4016)	07C9	CB 19	RR C
0774	A7	AND A	07CB	38 FA	JR C,07C7
0775	ED 42	SBC HL,BC	07CD	0C	INC C
0777	20 03	JR NZ,077C	07CE	C0	RET NZ
0779	3E B8	LD A,B8	07CF	48	LD C,B
077B	D7	RST 10	07D0	2D	DEC L
077C	2A 16 40	LD HL,(4016)	07D1	2E 01	LD L,01
077F	7E	LD A,(HL)	07D3	20 F2	JR NZ,07C7
0780	23	INC HL	07D5	21 7D 00	LD HL,007D
0781	CD B4 07	CALL 07B4	07D8	5F	LD E,A
0784	22 16 40	LD (4016),HL	07D9	19	ADD HL,DE
0787	28 E4	JR Z,076D	07DA	37	SCF
0789	FE 7F	CP 7F	07DB	C9	RET
078B	28 10	JR Z,079D	07DC	7B	LD A,E
078D	FE 76	CP 76	07DD	A7	AND A
078F	28 5D	JR Z,07EE	07DE	F8	RET M
0791	CB 77	BIT 6,A	07DF	18 10	JR 07F1
0793	28 05	JR Z,079A	07E1	AF	XOR A
0795	CD 4B 09	CALL 094D	07E2	09	ADD HL, BC

Ind.	Codice	Assembler
07E3	3C	INC A
07E4	38 FC	JR C,07E2
07E6	ED 42	SBC HL,BC
07E8	3D	DEC A
07E9	28 F1	JR Z,07DC
07EB	1E 1C	LD E,1C
07ED	83	ADD A,E
07EE	A7	AND A
07EF	28 04	JR Z,07F5
07F1	FD CB 01 66	RES 0,(IY+01)
07F5	D9	EXX
07F6	E5	PUSH HL
07F7	FD CB 01 4E	BIT 1,(IY+01)
07FB	20 05	JR NZ,0802
07FD	CD 08 08	CALL 0808
0800	18 03	JR 0805
0802	CD 51 08	CALL 0851
0805	E1	POP HL
0806	D9	EXX
0807	C9	RET
0808	57	LD D,A
0809	ED 4B 39 40	LD BC,(4039)
080D	79	LD A,C
080E	FE 21	CP 21
0810	28 1A	JR Z,0B2C
0812	3E 76	LD A,76
0814	BA	CP D
0815	28 30	JR Z,0847
0817	2A 0E 40	LD HL,(400E)
081A	BE	CP (HL)
081B	7A	LD A,D
081C	20 20	JR NZ,083E
081E	0D	DEC C
081F	20 19	JR NZ,083A
0821	23	INC HL
0822	22 0E 40	LD (400E),HL
0825	0E 21	LD C,21
0827	05	DEC B
0828	ED 43 39 40	LD (4039),BC
082C	78	LD A,B
082D	FD BE 22	CP (IY+22)
0830	28 03	JR Z,0835
0832	A7	AND A
0833	20 DD	JR NZ,0812
0835	2E 04	LD L,04
0837	C3 58 00	JP 0058
083A	CD 9B 09	CALL 099B
083D	EB	EX DE,HL
083E	77	LD (HL),A
083F	23	INC HL

Ind.	Codice	Assembler
0840	22 0E 40	LD (400E),HL
0843	FD 35 39	DEC (IY+39)
0846	C9	RET
0847	0E 21	LD C,21
0849	05	DEC B
084A	FD CB 01 C6	SET 0,(IY+01)
084E	C3 18 09	JP 0918
0851	FE 76	CP 76
0853	28 1C	JR Z,0871
0855	4F	LD C,A
0856	3A 38 40	LD A,(4038)
0859	E6 7F	AND 7F
085B	FE 5C	CP 5C
085D	6F	LD L,A
085E	26 40	LD H,40
0860	CC 71 08	CALL Z,0871
0863	71	LD (HL),C
0864	2C	INC L
0865	FD 75 38	LD (IY+38),L
0868	C9	RET
0869	16 16	LD D,16
086B	2A 0C 40	LD HL,(400C)
086E	23	INC HL
086F	18 05	JR 0876
0871	16 01	LD D,01
0873	21 3C 40	LD HL,403C
0876	CD E7 02	CALL 02E7
0879	C5	PUSH BC
087A	E5	PUSH HL
087B	AF	XOR A
087C	5F	LD E,A
087D	D3 FB	OUT (FB),A
087F	E1	POP HL
0880	CD 46 0F	CALL 0F46
0883	38 05	JR C,088A
0885	1F	RRA
0886	D3 FB	OUT (FB),A
0888	CF	RST 8
0889	0C	INC C
088A	DB FB	IN A,(FB)
088C	87	ADD A,A
088D	FA DE 08	JP M,08DE
0890	30 EE	JR NC,0880
0892	E5	PUSH HL
8093	D5	PUSH DE
0894	7A	LD A,D
0895	FE 02	CP 02
0897	9F	SBC A,A
0898	A3	AND E
0899	07	RLCA

Ind.	Codice	Assembler
089A	A3	AND E
089B	57	LD D,A
089C	4E	LD C,(HL)
089D	79	LD A,C
089E	23	INC HL
089F	FE 76	CP 76
08A1	28 24	JR Z,(08C7)
08A3	E5	PUSH HL
08A4	CB 27	SLA A
08A6	87	ADD A,A
08A7	87	ADD A,A
08A8	26 0F	LD H,0F
08AA	CB 14	RL H
08AC	83	ADD A,E
08AD	6F	LD L,A
08AE	CB 11	RL C
08B0	9F	SBC A,A
08B1	AE	XOR (HL)
08B2	4F	LD C,A
08B3	06 08	LD B,08
08B5	7A	LD A,D
08B6	CB 01	RLC C
08B8	1F	RRA
08B9	67	LD H,A
08BA	DB FB	IN A,(FB)
08BC	1F	RRA
08BD	30 FB	JR NC,08BA
08BF	7C	LD A,H
08C0	D3 FB	OUT (FB),A
08C2	10 F1	DJNZ 08B5
08C4	E1	POP HL
08C5	18 D5	JR 089C
08C7	DB FB	IN A,(FB)
08C9	1F	RRA
08CA	30 FB	JR NC,08C7
08CC	7A	LD A,D
08CD	0F	RRCA
08CE	D3 FB	OUT (FB),A
08D0	D1	POP DE
08D1	1C	INC E
08D2	CB 5B	BIT 3,E
08D4	28 A7	JR Z,087D
08D6	C1	POP BC
08D7	15	DEC D
08D8	20 A0	JR NZ,087A
08DA	3E 04	LD A,0544
08DC	D3 FB	OUT (FB),A
08DE	CD 07 02	CALL 0207
08E1	C1	POP BC
08E2	21 5C 40	LD HL,405C
08E5	36 76	LD (HL),76
08E7	06 20	LD B,20
08E9	2B	DEC HL
08EA	36 00	LD (HL),00
08EC	10 FB	DJNZ 08E9
08EE	7D	LD A,L
08EF	CB FF	SET 7,A
08F1	32 38 40	LD (4038),A
08F4	C9	RET
08F5	3E 17	LD A,17
08F7	90	SUB B
08F8	38 0B	JR C,0905
08FA	FD BE 22	CP (IY+22)
08FD	DA 35 08	JP C,0835
0900	3C	INC A
0901	47	LD B,A
0902	3E 1F	LD A,1F
0904	91	SUB C
0905	DA AD 0E	JP C,0EAD
0908	C6 02	ADD A,02
090A	4F	LD C,A
090B	FD CB 01 4E	BIT 1,(IY+01)
090F	28 07	JR Z,0918
0911	3E 5D	LD A,5D
0913	91	SUB C
0914	32 38 40	LD (4038),A
0917	C9	RET
0918	ED 43 39 40	LD (4039),BC
091C	2A 10 40	LD HL,(4010)
091F	51	LD D,C
0920	3E 22	LD A,22
0922	91	SUB C
0923	4F	LD C,A
0924	3E 76	LD A,76
0926	04	INC B
0927	2B	DEC HL
0928	BE	CP (HL)
0929	20 FC	JR NZ,0927
092B	10 FA	DJNZ 0927
092D	23	INC HL
092E	ED B1	CPIR
0930	2B	DEC HL
0931	22 0E 40	LD (400E),HL
0934	37	SCF
0935	E0	RET PO
0936	15	DEC D
0937	C8	RET Z
0938	C5	PUSH BC
0939	CD 9E 09	CALL 099E
093C	C1	POP BC

Ind.	Codice	Assembler
093D	41	LD B,C
093E	62	LD H,B
093F	6B	LD L,E
0940	36 00	LD (HL),00
0942	2B	DEC HL
0943	10 FB	DJNZ 0940
0945	EB	EX DE,HL
0946	23	INC HL
0947	22 0E 40	LD (400E),HL
094A	C9	RET
094B	F5	PUSH AF
094C	CD 75 09	CALL 0975
094F	30 08	JR NC,0959
0951	FD CB 01 46	BIT 0,(IY+01)
0955	20 02	JR NZ,0959
0957	AF	XOR A
0958	D7	RST 10
0959	0A	LD A,(BC)
095A	E6 3F	AND 3F
095C	D7	RST 10
095D	0A	LD A,(BC)
095E	03	INC BC
095F	87	ADD A,A
0960	30 F7	JR NC,0959
0962	C1	POP BC
0963	CB 78	BIT 7,B
0965	C8	RET Z
0966	FE 1A	CP 1A
0968	28 03	JR Z,096D
096A	FE 38	CP 38
096C	D8	RET C
096D	AF	XOR A
096E	FD CB 01 C6	SET 0,(IY+01)
0972	C3 F5 07	JP 07F5
0975	E5	PUSH HL
0976	21 11 01	LD HL,0111
0979	CB 7F	BIT 7,A
097B	28 02	JR Z,097F
097D	E6 3F	AND 3F
097F	FE 43	CP 43
0981	30 10	JR NC,0993
0983	47	LD B,A
0984	04	INC B
0985	CB 7E	BIT 7,(HL)
0987	23	INC HL
0988	28 FB	JR Z,0905
098A	10 F9	DJNZ 0985
098C	CB 77	BIT 6,A
098E	20 02	JR NZ,0992
0990	FE 18	CP 18
0992	3F	CCF
0993	44	LD B,H
0994	4D	LD C,L
0995	E1	POP HL
0996	D0	RET NC
0997	0A	LD A,(BC)
0998	C6 E4	ADD A,E4
099A	C9	RET
099B	01 01 00	LD BC,0001
099E	E5	PUSH HL
099F	CD C5 0E	CALL 0EC5
09A2	E1	POP HL
09A3	CD AD 09	CALL 09AD
09A6	2A 1C 40	LD HL,(401C)
09A9	EB	EX DE,HL
09AA	ED B8	LDDR
09AC	C9	RET
09AD	F5	PUSH AF
09AE	E5	PUSH HL
09AF	21 0C 40	LD HL,400C
09B2	3E 09	LD A,09
09B4	5E	LD E,(HL)
09B5	23	INC HL
09B6	56	LD D,(HL)
09B7	E3	EX (SP),HL
09B8	A7	AND A
09B9	ED 52	SBC HL,DE
09BB	19	ADD HL,DE
09BC	E3	EX (SP),HL
09BD	30 09	JR HC,09CB
09BF	D5	PUSH DE
09C0	EB	EX DE,HL
09C1	09	ADD HL, BC
09C2	EB	EX DE,HL
09C3	72	LD (HL),D
09C4	2B	DEC HL
09C5	73	LD (HL),E
09C6	23	INC HL
09C7	D1	POP DE
09C8	23	INC JL
09C9	3D	DEC A
09CA	20 E8	JR HZ,09B4
09CC	EB	EX DE,HL
09CD	D1	POP DE
09CE	F1	POP AF
09CF	A7	AND A
09D0	ED 52	SBC HL,DE
09D2	44	LD B,H
09D3	4D	LD C,L
09D4	03	INC BC

Ind.	Codice	Assembler
09D5	19	ADD HL,DE
09D6	EB	EX DE,HL
09D7	C9	RET
09D8	E5	PUSH HL
09D9	21 7D 40	LD HL,407D
09DC	54	LD D,H
09DD	5D	LD E,L
09DE	C1	POP BC
09DF	CD EA 09	CALL 09EA
09E2	D0	RET NC
09E3	C5	PUSH BC
09E4	CD F2 09	CALL 09F2
09E7	EB	EX DE,HL
09E8	18 F4	JR 09DE
09EA	7E	LD A,(HL)
09EB	B8	CP B
09EC	C0	RET NZ
09ED	23	INC HL
09EE	7E	LD A,(HL)
09EF	2B	DEC HL
09F0	B9	CP C
09F1	C9	RET
09F2	E5	PUSH HL
09F3	7E	LD A,(HL)
09F4	FE 40	CP 40
09F6	38 17	JR C,0A0F
09F8	CB 6F	BIT 5,A
09FA	28 14	JR Z,0A10
09FC	87	ADD A,A
09FD	FA 01 0A	JP M,0A01
0A00	3F	CCF
0A01	01 05 00	LD BC,0005
0A04	30 02	JR NC,0A0B
0A06	0E 11	LD C,11
0A08	17	RLA
0A09	23	INC HL
0A0A	7E	LD A,(HL)
0A0B	30 FB	JR NC,0A08
0A0D	18 06	JR 0A15
0A0F	23	INC HL
0A10	23	INC HL
0A11	4E	LD C,(HL)
0A12	23	INC HL
0A13	46	LD B,(HL)
0A14	23	INC HL
0A15	09	ADD HL,BC
0A16	D1	POP DE
0A17	A7	AND A
0A18	ED 52	SBC HL,DE
0A1A	44	LD B,H

Ind.	Codice	Assembler
0A1B	4D	LD C,L
0A1C	19	ADD HL,DE
0A1D	EB	EX DE,HL
0A1E	C9	RET
0A1F	FD 46 22	LD B,(IY+22)
0A22	C5	PUSH BC
0A23	CD 2C 0A	CALL 0A2C
0A26	C1	POP BC
0A27	05	DEC B
0A28	18 02	JR 0A2C
0A2A	06 18	LD B,18
0A2C	FD CB 01 8E	RES 1,(IY+01)
0A30	0E 21	LD C,21
0A32	C5	PUSH BC
0A33	CD 18 09	CALL 0918
0A36	C1	POP BC
0A37	3A 05 40	LD A,(4005)
0A3A	FE 4D	CP 4D
0A3C	38 14	JR C,0A52
0A3E	FD CB 3A FE	SET 7,(IY+3A)
0A42	AF	XOR A
0A43	CD F5 07	CALL 07F5
0A46	2A 39 40	LD HL,(4039)
0A49	7D	LD A,L
0A4A	B4	OR H
0A4B	E6 7E	AND 7E
0A4D	20 F3	JR NZ,0A42
0A4F	C3 18 09	JP 0918
0A52	54	LD D,H
0A53	5D	LD E,L
0A54	2B	DEC HL
0A55	48	LD C,B
0A56	06 00	LD B,00
0A58	ED B0	LDIR
0A5A	2A 10 40	LD HL,(4010)
0A5D	CD 17 0A	CALL 0A17
0A60	C5	PUSH BC
0A61	78	LD A,B
0A62	2F	CPL
0A63	47	LD B,A
0A64	79	LD A,C
0A65	2F	CPL
0A66	4F	LD C,A
0A67	03	INC BC
0A68	CD AD 09	CALL 09AD
0A6B	EB	EX DE,HL
0A6C	E1	POP HL
0A6D	19	ADD HL,DE
0A6E	D5	PUSH DE
0A6E	ED B0	LDIR

Ind.	Codice	Assembler
0A71	E1	POP HL
0A72	C9	RET
0A73	2A 14 40	LD HL,(4014)
0A76	CD 4D 00	CALL 004D
0A79	DF	RST 18
0A7A	FD CB 2D 6E	BIT 5,(IY+2D)
0A7E	C0	RET NZ
0A7F	21 5D 40	LD HL,405D
0A82	22 1C 40	LD (401C),HL
0A85	CD 48 15	CALL 1548
0A88	CD 8A 15	CALL 158A
0A8B	38 04	JR C,0A91
0A8D	21 F0 D8	LD HL,D8F0
0A90	09	ADD HL, BC
0A91	DA 9A 0D	JP C,0D9A
0A94	BF	CP A
0A95	C3 BC 14	JP 14BC
0A98	D5	PUSH DE
0A99	E5	PUSH HL
0A9A	AF	XOR A
0A9B	CB 78	BIT 7,B
0A9D	20 20	JR NZ,0ABF
0A9F	60	LD H,B
0AA0	69	LD L,C
0AA1	1E FF	LD E,FF
0AA3	18 08	JR 0AAD
0AA5	D5	PUSH DE
0AA6	56	LD D,(HL)
0AA7	23	INC HL
0AA8	5E	LD E,(HL)
0AA9	E5	PUSH HL
0AAA	EB	EX DE,HL
0AAB	1E 00	LD E,00
0AAD	01 18 FC	LD BC,FC18
0AB0	CD E1 07	CALL 07E1
0AB3	01 9C FF	LD BC,FF9C
0AB6	CD E1 07	CALL 07E1
0AB9	0E F6	LD C,F6
0ABB	CD E1 07	CALL 07E1
0ABE	7D	LD A,L
0ABF	CD EB 07	CALL 07EB
0AC2	E1	POP HL
0AC3	D1	POP DE
0AC4	C9	RET
0AC5	CD A6 0D	CALL 0DA6
0AC8	E1	POP HL
0AC9	C8	RET Z
0ACA	E9	JP (HL)
0ACB	FD CB 01 CE	SET 1,(IY+01)
0ACF	7E	LD A,(HL)
0AD0	FE 76	CP 76
0AD2	CA 84 0B	JP Z,0B84
0AD5	D6 1A	SUB 1A
0AD7	CE 00	ADC A,00
0AD9	28 69	JR Z,0B44
0ADB	FE A7	CP A7
0ADD	20 1B	JR NZ,0AFA
0ADF	E7	RST 20
0AE0	CD 92 0D	CALL 0D92
0AE3	FE 1A	CP 1A
0AE5	C2 9A 0D	JP NZ,0D9A
0AE8	E7	RST 20
0AE9	CD 92 0D	CALL 0D92
0AEC	CD 4E 0B	CALL 0B4E
0AEF	EF	RST 28
0AF0	01 34 CD	LD BC,CD34
0AF3	F5	PUSH AF
0AF4	0B	DEC BC
0AF5	CD F5 08	CALL 08F5
0AF8	18 3D	JR 0B37
0AFA	FE A8	CP A8
0AFC	20 33	JR NZ,0B31
0AFE	E7	RST 20
0AFF	CD 92 0D	CALL 0D92
0B02	CD 4E 0B	CALL 0B4E
0B05	CD 02 0C	CALL 0C02
0B08	C2 AD 0E	JP NZ,0EAD
0B0B	E6 1F	AND 1F
0B0D	4F	LD C,A
0B0E	FD CB 01 4E	BIT 1,(IY+01)
0B12	28 0A	JR Z,0B1E
0B14	FD 96 38	SUB (IY+38)
0B17	CB FF	SET 7,A
0B19	C6 3C	ADD A,3C
0B1B	D4 71 08	CALL NC,0871
0B1E	FD 86 39	ADD A,(IY+39)
0B21	FE 21	CP 21
0B23	3A 3A 40	LD A,(403A)
0B26	DE 01	SBC A,01
0B28	CD FA 08	CALL 08FA
0B2B	FD CB 01 C6	SET 0,(IY+01)
0B2F	18 06	JR 0B37
0B31	CD 55 0F	CALL 0F55
0B34	CD 55 0B	CALL 0B55
0B37	DF	RST 18
0B38	D6 1A	SUB 1A
0B3A	CE 00	ADC A,00
0B3C	28 06	JR Z,0B44

Ind.	Codice	Assembler
0B3E	CD 1D 0D	CALL 0D1D
0B41	C3 84 0B	JP 0B84
0B44	D4 8B 0B	CALL NC,0B8B
0B47	E7	RST 20
0B48	FE 76	CP 76
0B4A	C8	RET Z
0B4B	C3 D5 0A	JP 0AD5
0B4E	CD A6 0D	CALL 0DA6
0B51	C0	RET NZ
0B52	E1	POP HL
0B53	18 E2	JR 0B37
0B55	CD C5 0A	CALL 0AC5
0B58	FD CB 01 76	BIT 6,(IY+01)
0B5C	CC F8 13	CALL Z,13F8
0B5F	28 0A	JR Z,0B6B
0B61	C3 DB 15	JP 15DB
0B64	3E 0B	LD A,0B
0B66	D7	RST 10
0B67	ED 5B 18 40	LD DE,(4018)
0B6B	78	LD A,B
0B6C	B1	OR C
0B6D	0B	DEC BC
0B6E	C8	RET Z
0B6F	1A	LD A,(DE)
0B70	13	INC DE
0B71	ED 53 18 40	LD (4018),DE
0B75	CB 77	BIT 6,A
0B77	28 ED	JR Z,0B66
0B79	FE C0	CP C0
0B7B	28 E7	JR Z,0B64
0B7D	C5	PUSH BC
0B7E	CD 4B 09	CALL 094B
0B81	C1	POP BC
0B82	18 E3	JR 0B67
0B84	CD C5 0A	CALL 0AC5
0B87	3E 76	LD A,76
0B89	D7	RST 10
0B8A	C9	RET
0B8B	CD C5 0A	CALL 0AC5
0B8E	FD CB 01 C6	SET 0,(IY+01)
0B92	AF	XOR A
0B93	D7	RST 10
0B94	ED 4B 39 40	LD BC,(4039)
0B98	79	LD A,C
0B99	FD CB 01 4E	BIT 1,(IY+01)
0B9D	28 05	JR Z,0BA4
0B9F	3E 5D	LD A,5D
0BA1	FD 96 38	SUB (IY+38)
0BA4	0E 11	LD C,11
0BA6	B9	CP C
0BA7	30 02	JR NC,0BAB
0BA9	0E 01	LD C,01
0BAB	CD 0B 09	CALL 090B
0BAE	C9	RET
0BAF	CD F5 0B	CALL 0BF5
0BB2	ED 43 36 40	LD (4036),BC
0BB6	3E 2B	LD A,2B
0BB8	90	SUB B
0BB9	DA AD 0E	JP C,0EAD
0BBC	47	LD B,A
0BBD	3E 01	LD A,01
0BBF	CB 28	SRA B
0BC1	30 02	JR NC,0BC5
0BC3	3E 04	LD A,04
0BC5	CB 29	SRA C
0BC7	30 01	JR NC,0BCA
0BC9	07	RLCA
0BCA	F5	PUSH AF
0BCB	CD F5 08	CALL 08F5
0BCE	7E	LD A,(HL)
0BCF	07	RLCA
0BD0	FE 10	CP 10
0BD2	30 06	JR NC,0BDA
0BD4	0F	RRCA
0BD5	30 02	JR HC,0BD9
0BD7	EE 8F	XOR 8F
0BD9	47	LD B,A
0BDA	11 9E 0C	LD DE,0C9E
0BDD	3A 30 40	LD A,(4030)
0BE0	93	SUB E
0BE1	FA E9 0B	JP M,0BE9
0BE4	F1	POP AF
0BE5	2F	CPL
0BE6	A0	AND B
0BE7	18 02	JR 0BEB
0BE9	F1	POP AF
0BEA	B0	OR B
0BEB	FE 08	CP 08
0BED	38 02	JR C,0BF1
0BEF	EE 8F	XOR 8F
0BF1	D9	EXX
0BF2	D7	RST 10
0BF3	D9	EXX
0BF4	C9	RET
0BF5	CD 02 0C	CALL 0C02
0BF8	47	LD B,A
0BF9	C5	PUSH BC
0BFA	CD 02 0C	CALL 0C02
0BFD	59	LD E,C
0BFE	C1	POP BC

```
Ind. Codice      Assembler      Ind. Codice      Assembler

0BFF 51          LD D,C         0C13 CD 18 09    CALL 0918
0C00 4F          LD C,A         0C16 CD 9B 09    CALL 099B
0C01 C9          RET            0C19 7E          LD A,(HL)
0C02 CD CD 15    CALL 15CD      0C1A 12          LD (DE),A
0C05 DA AD 0E    JP C,0EAD      0C1B FD 34 3A    INC (IY+3A)
0C08 0E 01       LD C,01        0C1E 2A 0C 40    LD HL,(400C)
0C0A C8          RET Z          0C21 23          INC HL
0C0B 0E FF       LD C,FF        0C22 54          LD D,H
0C0D C9          RET            0C23 5D          LD E,L
0C0E FD 46 22    LD B,(IY+22)   0C24 ED B1       CPIR
0C11 0E 21       LD C,21        0C26 C3 5D 0A    JP 0A5D
```

Si riportano gli indirizzi di inizio di alcune routine:

Ind. esadec.	Ind. decim.	Funzione
01FC	508	Usata dalle routine dei comandi LOAD e SAVE.
0207	519	Invio fotogrammi al video.
02BB	699	Scansione tastiera.
02F6	758	SAVE.
0340	832	LOAD.
03CB	971	Usata per l'inizializzazione del del sistema e dopo il comando NEW.
03E5	997	Routine principale per inizializ= zare il sistema.
063E	1598	RUN.
07B4	1972	Decodifica tasti, il valore del tasto sta nei registri BC.
07F1	2033	Stampa caratteri, da RST 0010.
08F5	2293	Espansione display file quando non e' mappato in memoria.
0A2A	2602	CLS.
0ACF	2767	PRINT.
0BAF	2992	PLOT/UNPLOT.

0C0E	3086	SCROLL.
0CBA	3258	Inizio interprete Basic.
0F20	3872	FAST.
0F28	3879	SLOW.
131D	4893	LET.
1405	5125	DIM.
14CA	5322	Trattamento numeri in forma espo= nenziale.
1914	6420	Tabella delle funzioni.
199C	6556	Calcoli.
1AA9	6825	Sviluppo funzioni.

La tabella per la generazione dei caratteri si trova da 7680 a 8191.

Con il programma che segue si possono listare parti di codice macchina fornendo l'indirizzo di inizio e l'indirizzo finale in decimale. Il programma lista sulla stampante 9 byte per riga in esadecimale, scrivendo all'inizio della linea l'indirizzo esadecimale del primo byte. Prima del blocco dei dati lista l'indirizzo di inizio e di fine in decimale. Se il numero dei byte richiesti non e' multiplo di 9 ne vengono listati alcuni in piu'.

```
 2 INPUT N1
 4 INPUT N2
 5 LPRINT N1,N2
 6 LPRINT
10 FOR K=N1 TO N2 STEP 9
15 LET X=INT(K/4096)
20 LET Y=K-X*4096
25 LET Z=INT(Y/256)
30 LET Y=Y-Z*256
35 LET T=INT(Y/16)
40 LET Y=Y-T*16
50 LPRINT CHR$(X+28);CHR$(Z+28);CHR$(T+28);CHR$(Y+28);
60 FOR I=1 TO 9
65 LET X=PEEK(K+I-1)
67 LET Y=INT(X/16)
68 LET Z=X-Y*16
69 LPRINT " ";CHR$(Y+28);CHR$(Z+28);
```

```
70 NEXT I
80 LPRINT
90 NEXT K
```

INDICE ANALITICO

L. 16.500 Cod. 318B Cod. ISBN 88-7056-107-0

La dr. Rita Bonelli, laureata in Matematica e Fisica presso l'Università di Milano, può vantare un'esperienza di circa 25 anni nell'analisi dei sistemi organizzativi e nella programmazione dei calcolatori elettronici.
Per più di dieci anni ha affiancato alle attività professionali le attività didattiche, come titolare di una cattedra di informatica presso l'Istituto Tecnico Industriale Feltrinelli di Milano.
Attualmente si interessa anche di mini e personal computers, studiando il software applicativo per particolari categorie di utenti, e tenendo corsi di programmazione a vari livelli.

Il testo abbraccia tre calcolatori: lo ZX81, lo ZX80 e lo ZX80 Nuova ROM, che, seppur filosoficamente equivalenti, presentano notevoli differenze nel sistema di gestione e, soprattutto, nel BASIC usato. Li confronta tra loro, traendone quindi quelle necessarie considerazioni sulla loro potenzialità e limiti nell'ambito di ciascun contesto.
Alcune parti di questo libro derivano direttamente dal precedente "Impariamo a programmare in BASIC con lo ZX80", purgate ed ampliate però, alla luce di quelle che sono state (e tante) le richieste di ulteriori approfondimenti su argomenti specifici quali: trasformazione dei programmi da un calcolatore all'altro, sistema operativo, gestione dei file, linguaggio macchina. L'agile testo originale evolve, così, in questa guida che pur mantenendo chiarezza e semplicità espositiva e ricchezza di esemplificazione, risulta ora un vero e proprio strumento operativo per tutti coloro che vogliono imparare l'informatica in generale e la programmazione in BASIC in particolare, travalicando gli stessi sistemi esposti.
Partendo da quello che è un computer, il lettore impara nei primi sei capitoli a programmare in BASIC. Con i capitoli 7 e 8 si spinge oltre: all'utilizzo della memoria e al linguaggio macchina.
Nel capitolo 9 sono contenuti, poi, parecchi programmi, e per ciascuno vengono fornite, dove possibile, le diverse versioni. Sempre in questo capitolo si parla di file e di animazione delle figure. E per finire ben otto Appendici, essenziali ed utilissime tra cui spiccano le due dedicate ai sistemi operativi dello ZX80 e ZX81.

GRUPPO
EDITORIALE
JACKSON

Rita Bonelli

GUIDA AL SINCLAIR ZX81 ZX80 e Nuova ROM

318B